People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University Akli Mohand Oulhadj of Bouira
Faculty of Sciences and Applied Sciences
Computer Science Department

Master Thesis

Specialty: *Computer Systems Engineering*

Theme

Designing and implementation of smart contracts system for Internet of things application

Supervised by:

– *Mr. DJELLABI Brahim*

Realized by:

– *BOUDISSA Dahmane Lyes*
– *MOHAMMED MERABET Mohamed Amine*

Discussed by:

– ***President:*** *Mr. ABBAS Akli*
– ***Examiner 1:*** *Mr. BAL Kamal*
– ***Examiner 2:*** *Mr. BADIS Lyes*

2020/2019

Thanks

First of all we thank the almighty ALLAH for given us the courage and the patience to realize this modest work. We would like to express our sincere gratitude to all those who contributed directly or indirectly to the completion of this thesis.

A special and sincere thank you to our parents for the efforts provided.

Our thanks go in particular to Our promoter Mr. DJELLABI Brahim who supervised ,and for his support, his sound advice and his great courage during the development of this work.

A thanks to our friends who supported us during this thesis.

Our thanks also go to all the pedagogical team of the IT department of UAMOB.

Dedicate

We dedicate this modest work:

To our dear parents, for all their sacrifices, their love, their tenderness, their support and their prayers throughout our studies.

To our dear sisters and my fiancée H.H, for their permanent encouragement and moral support.

To all our family for their support throughout of our university career, To all our friends and colleagues.

To all students of the 2020/2019 class: Computer Science, To all those who have helped and supported us from near and far, either by a word or by a gesture.

Thank you for always being there for us.

Mohammed Merabet Mohamed Amine

Dedicate

I dedicate this modest work to after thanking ALLAH for all the blessings and :

To my mother who encouraged and gave me all her love and support.

To the man who guided me in this life to get my success : my father.

To my sisters Roumaissa and Lamia and her little angel Rovan.

To all my close friends who supported me and believe in me during university,without forgetting my close B.H.

To all my family for being supportive.

Thank you a lot.

Boudissa Dahmane Lyes

Table of Contents

Chapter III : "Milk supply chain, Implementation and Comparison"

List of Figures

Liste of Tables

List of Abbreviation

P2P	Peer-to-Peer
RAM	Random access memory
PoW	Proof of Work
Asic	application-specific integrated circuit
PoS	Proof of Stake
DPoS	Delegated Proof of Stake
PoD	Proof of Deposit
BFT	Byzantine Fault tolerance
PBFT	Practical Byzantine Fault Tolerance
SSL	Secure Sockets Layer
D-Apps	Decentralized applications
ICOs	Initial Coin Offerings
EMR	Electronic Medical Record
IoT	Internet Of Things
KYC	Knowing your client
AML	Antimony laundering
SCM	Supply chain management
ERP	Enterprise Resource Planning
B2B	Business-to-Business
AAA	Authentication, Authorization and Accounting
PSTN	Public switched telephone network
DSL	Digital subscriber lines
RFID	Radio Frequency Identifications
UWB	Ultra- broadband
NFC	Near Field Communication
LPWAN	Low power wireless area network
LoRa	Long-range network
GSM	Global System for Mobile Communications
CDMA	Code Division Multiple Access
LTE	Long-Term Evolution
AMQP	Advanced Message Queuing Protocol
TLS	Transport Layer Security
DLT	Distributed ledger Technology
UTXO	Unspent transaction output
CLI	Command line interface

General introduction

Recently, Cryptocurrency and Blockchain are among the advanced technologies widely used thanks to the progress of research in distributed system security. Likewise, the realm of the Internet of things is knowing a significant increase owing to the number of pervasive devices connected to the internet, and the wealth of data generated by these devices. On another hand, the growth of the economy and industry has raised the life level of many people across the world. However, The number of parties involved in this economy is highly increasing as well. Moreover, the need for trusted intermediators, arbitrations, and enforcement costs, fraud losses make business transactions and the complexity of trading procedures a real challenge. One of these challenges is related to the traditional supply chain. Most supply chains in the industry rely mainly on the manual process, which makes the supply chain difficult to operate and time-consuming while tracking back the product. For example, what happened recently. E.coli romaine lettuce problem was responsible for at 200 getting sick, similarly, every year, unsafe food causes 600 million cases of foodborne disease. The manual process often takes up to a few weeks to trace pack the source. In this context, Supply chains based on smart contracts are found in the intersection between the Blockchain and the Internet of things, therefore, they can be considered the most suitable technologies that can cope with the burden of the manual processes in the ordinary supply chain.

Our work aims first, to develop a supply chain application based on smart contracts. Second, since the smart contract is a blockchain-based application, we opt to carry out a comparative study between private and public blockchain in terms of smart contracts and supply chain application performance. In this study presents a comparative analysis study on decentralized platforms Ethereum as a permission-less blockchain and Corda as permissioned blockchain. The example we have chosen as an application scenario is the dairy products supply chain. To achieve our goal, We begin with an introduction to distributed system and blockchain in the first chapter, in chapter 2, we present the concept of the internet of things, then, we introduce the concept of smart contracts and supply chain, the implementation, simulations, and validation the comparative study results are all discussed in the third chapter. Finally, we conclude and give some perspectives on our future work.

Chapter I

Decentralization, Distributed systems
P2P Network Overviews And Blockchain Technology

Introduction

In this chapter, and as an introduction to our research, we took a simple look at decentralized and distributed systems and peer-to-peer network as our first section preparation for entering The blockchain technology as the second section in which we will take a deep look and discuss it's structure and types and the important part the consensus mechanism in it and provide some examples (applications) used with this technology.

1. Decentralization

1.1. Definition

Decentralization is not a new concept. Strategy it been used in, management, and the government, for a long time. Distributing control and authority to the peripheries of an organization instead of one central being in full control of the organization and that is the basic idea of decentralization .the benefits of this configuration for organizations are increased efficiency, expedited decision making, better motivation, and a reduced burden on top management [17].

1.2. Decentralization methods

Two methods can be used to achieve decentralization:

1.2.1. Disintermediation

The concept of disintermediation can be explained with the aid of an example :

‘‘Imagine that you want to send money to a friend in another country. You go to a bank who charges you for a fee, will transfer your money to the bank in that country. For this situation, the bank keeps up a central database that is updated, confirming that you have sent the money’’.

Decentralization can be achieved by disintermediation, that’s by the elimination of intermediary, in case of sending money to your friend it's possible to send it directly no need to bank (intermediary) [3].

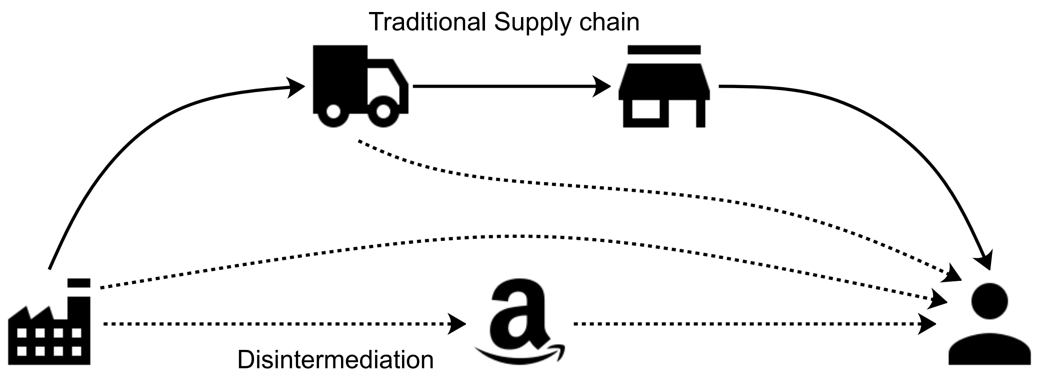


Figure 1.1 : Traditional supply chain vs disintermediation.

1.2.2. Decentralization driven by competition

In the method involving competition, in order to select the provision of services by the system, different service providers enter a competition with each other (based on their reputation, previous score, reviews, and quality of service), which allows smart contracts to make free choice on the criteria above, and the method are not going to show a full decentralization; In the following diagram, different level of decentralization, first is on the left where it shows a central system is in control, second on the right shows a fully decentralized is achieved by entirely remove intermediaries; In the center shows a semi decentralized Where Competing intermediaries or service providers are selected based on reputation or voting (achieving partial decentralization) [17].

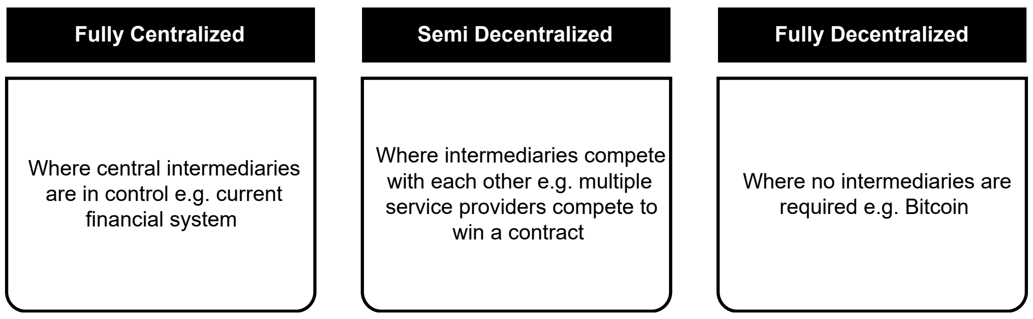

Fully Centralized
Where central intermediaries are in control e.g. current financial system
Semi Decentralized
Where intermediaries compete with each other e.g. multiple service providers compete to win a contract
Fully Decentralized
Where no intermediaries are required e.g. Bitcoin

Figure 1.2 : Scale of decentralization.

2. Distributed systems

2.1. Definition

Distributed systems are a model of computing that can achieve a common result where two or more nodes work together other in a coordinated way. Designed for a way that end users see it as a single logical platform. For example, Google's search engine relies on a big distributed system, but for the users, it looks like one coherent platform. All nodes are able to send and receive messages to and from each other. A Single node can be defined as a single player in a distributed system. The primary challenge in designing a distributed system is contract coordination and fault tolerance. Even if some nodes become corrupted or network links cuts, the distributed system must be able to tolerate this and continue working to achieve the desired result [17].

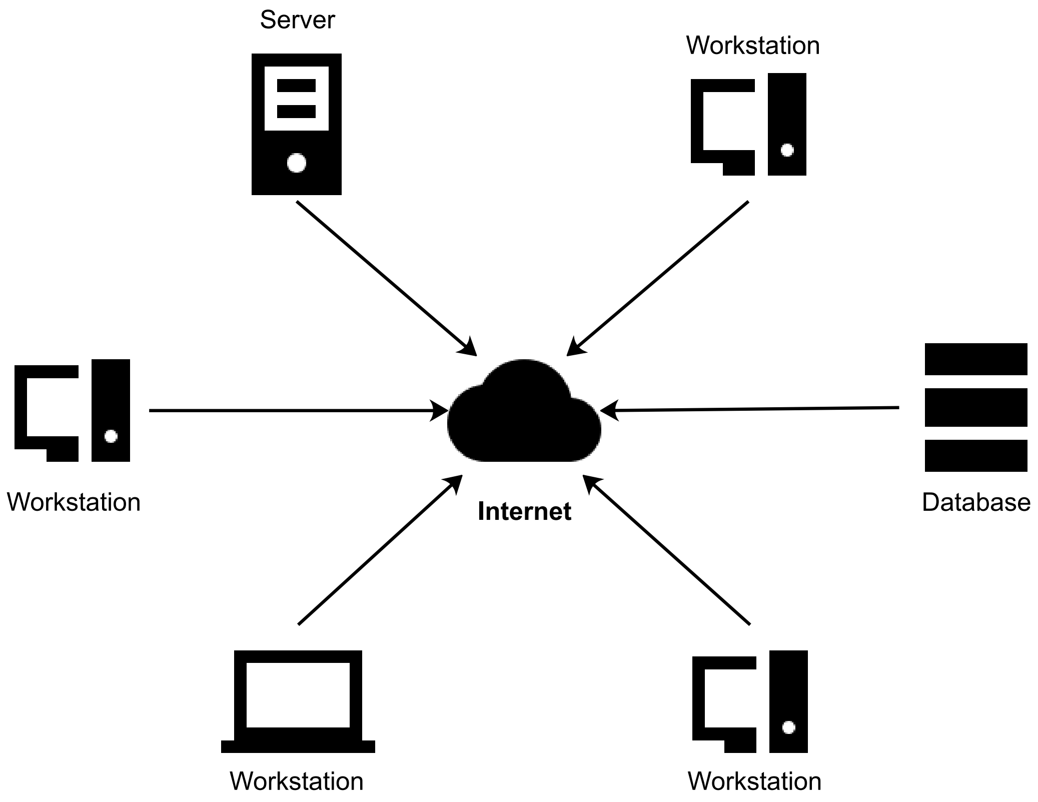


Figure 1.3 : Distributed systems network.

2.2. Advantages of distributed systems

The major advantages of a distributed system over single computers are:

2.2.1. Higher computing power

The computing power of a distributed system is the result of computing power of all connected computers combined. Subsequently, the compute power of distributed systems bigger than single computer [18]. This has been proven to be true even when comparing distributed systems contains relatively low computing power computers to isolated super computer.

2.2.2. Reduce costs

Over the past 20 years, the price of major computers, memory, disk space, and network equipment has decreased significantly. The initial costs for distributed systems are higher than the initial costs for individual computers, because distributed systems consist of many computers. However the costs of creating maintaining and operating a distributed system are less than the costs of creating, maintaining and operating an excellent computer [18]. This is especially true since replacing individual computers of a distributed system can be done without noticeable impact on the system.

2.2.3. Better reliability

The single super computer has less reliability than a distributed system. Because a distributed system is based on the fact that the entire network of computers can continue to operate even when individual devices crash. The distributed system does not have a one point of failure. If one item fails, the remaining items can take over. This what's increases reliability of distributed systems. [18]

2.2.4. Ability to normally grow

Can increase the computing power of the entire system by connecting additional computers to the system. Because the computing power of a distributed system is the result of the combined computing power of its components. As a result, the computing power of the entire system can be gradually increased over the fine-grained scale. This supports the way in which the demand for computing power increases in many organizations. The growing growth of distributed systems interferes with the growth of computing power for individual computers [18]. Individual computers provide identical power so till be replaced by a more powerful computer. This results in intermittent growth of computing power, rarely appreciated by consumers of computing services.

2.3. Disadvantages of distributed systems

The disadvantages of distributed systems compared to single computers are:

2.3.1. Overhead teamwork

Distributed systems do not have central entities that coordinate their members. Therefore, coordination must be carried out by the members of the system themselves [18]. Coordination work between co-workers in a distributed system is difficult and costs effort and computing power that cannot be spent on a real computing task.

2.3.2. Overhead communications

Coordination requires communication. Therefore, the computers that are in a distributed system must communicate with each other. This requires a communication protocol, sending, receiving, and processing messages, which in turn costs effort and computing power that cannot be spent on a real computing task. [18]

2.3.3. Dependencies on networks

Any type of communication requires support. Support is responsible for the transfer of information between the entities which communicate with each other. Computers communicate in distributed systems with messages that are transmitted over the network. Networks have their own challenges and tribulations, which in turn affect communication and coordination between the computers that are in the distributed system.[18] However, without any network, there would be no distributed system, no communication and therefore no coordination between the entities.

2.3.4. Complexity of higher programs

The complexity of the software increases .Because solving a computational problem include writing programs .Due to the mentioned disadvantages previously ,Any software in a distributed system should solve additional problems such as coordination, communication, and network use.[18]

2.3.5. Safety issues

Network communication means sending and sharing critical data for a real computing task. However, the transmission of information through a network carries security concerns because untrustworthy entities may misuse the network to access and exploit information.[18] Consequently, any distributed system has to address security concerns.

3. The difference between decentralization and distributed systems

Actually, numerous system depend on a central authority to direct the operation of network, rendering it rather central. For example, some P2P file-sharing systems permit clients to look and download files from different clients, yet they can't take a part in different processes, such as handling search queries; moreover, little networks operated by a restricted client base with common interest may also be said to have a higher level of centralization, in spite of the absence of centralized network foundation.[19]

The different types of systems that currently exist (central, decentralized, and distributed). And following diagrams shows them .This concept was first published by ***Paul Baran in On Distributed Communications: I. Introduction to Distributed Communications Networks :*** [19]

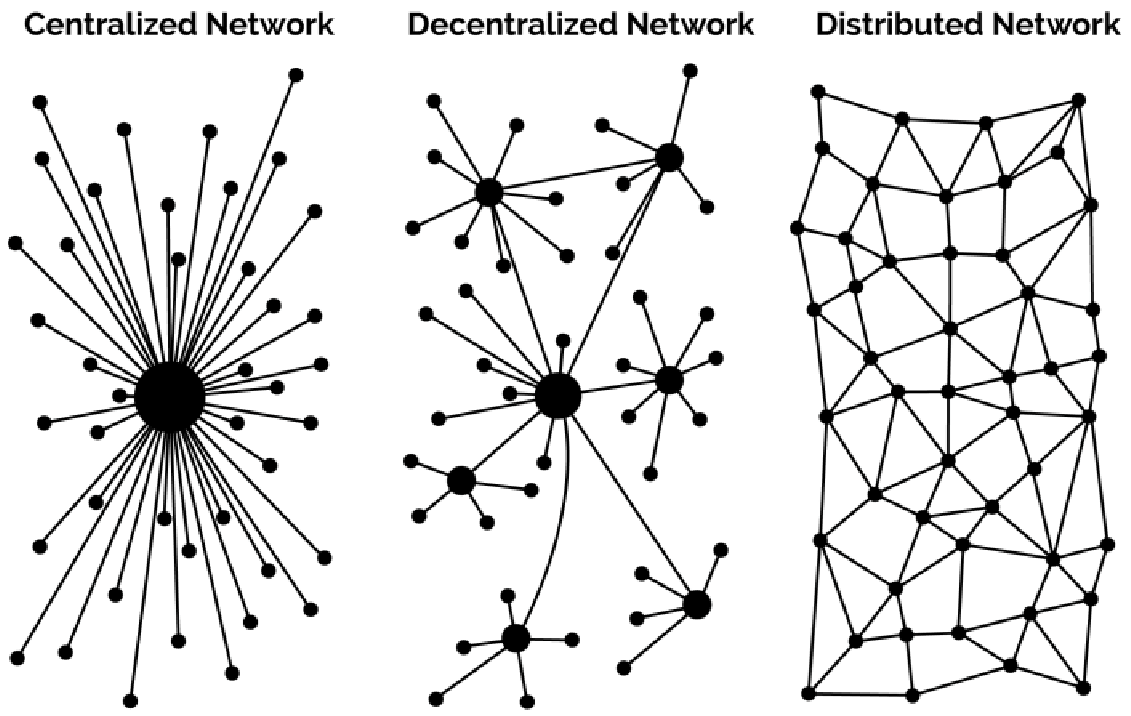


Figure 1.4 : Distributed communications networks. [19]

Centralized systems are traditional IT (client-server) systems in which there is one authority that controls the system, and is the only responsible for all operations on the system. All users of the centralized system depend on one service source. Distributed system, data and computation are spread on multiple nodes in the network. Sometimes this term is confused with parallel computing. Although there is some overlap in the definition, the main difference between these systems is that in a parallel computer system, the computation is performed by all the nodes simultaneously in order to obtain the result.

On the other hand, in a ***distributed system***, the computation may not occur in parallel and the data is replicated on several nodes which the users consider as a single and coherent system. Variations of these two models are used to achieve fault tolerance and speed. In a parallel system model, there is always a central authority that controls all nodes, which govern processing. This means that the system is by nature centralized. The critical difference between a decentralized system and a distributed system is that in a distributed system, there is still a central authority that governs the entire system.[19]

while in a ***decentralized system***, no such authority exists. The significant innovation in the decentralized model that led to this new era of decentralization of applications is a decentralized consensus. This mechanism got into play with Bitcoin, and it enabled the users to agree on something via a consensus algorithm without requiring a trusted third party, intermediary, or service provider.

4. P2P networks

4.1. Definition

A peer-to-peer (P2P) network in computer science is composed of collection of computers that store and exchange data collectively. Every node operates as a separate peer. All the nodes usually have equal power and perform the same tasks[16].

The term peer-to-peer in financial technology typically refers to cryptocurrencies or digital possessions through being exchanged over a distributed network. A P2P service allows sellers and buyers to carry out exchanges without mediators being required. Websites can also in some situations include a P2P platform linking buyers and suppliers.

P2P engineering can be ideal for different use cases, however it turned out to be especially well known during the 1990s when the first document-sharing projects were made. Today, the center of most cryptocurrencies are P2P network, making up an incredible part of the blockchain business[16].

In any case, they are additionally utilized in other conveyed processing applications, including web search engines, streaming services.

4.2. How does p2p work?

A global network of users operates P2P system. We typically will not have a central supervisor or server since each nodes hold a duplicate of the data-functioning to other nodes as both client and as server. Each node will import or upload files from other nodes. It distinct P2P network from the conventional client-server systems in which client computers download data from centralized server[16].

The linked devices share documents that are saved on their hard drives on P2P networks. Utilizing software applications intended to intervene the sharing of information, clients can ask different devices on the network to discover and download files. When a client has downloaded a specific file, they may then be able to serve as the source of that file.

Put it differently, when a node operates as a client, they download files from other system nodes. Yet while they are filling in as a server, they are the source that other nodes can download files from. Nevertheless, All tasks can be executed simultaneously (for example, downloading file A, and uploading file B)[16].

Because each node stores, transmits and gets data, P2P network will in general be quicker and more effective because their client base becomes bigger. The distributed structure also makes P2P networks highly vulnerable to cyber-attacks. In contrast to conventional models, P2P systems don't have a weak point.

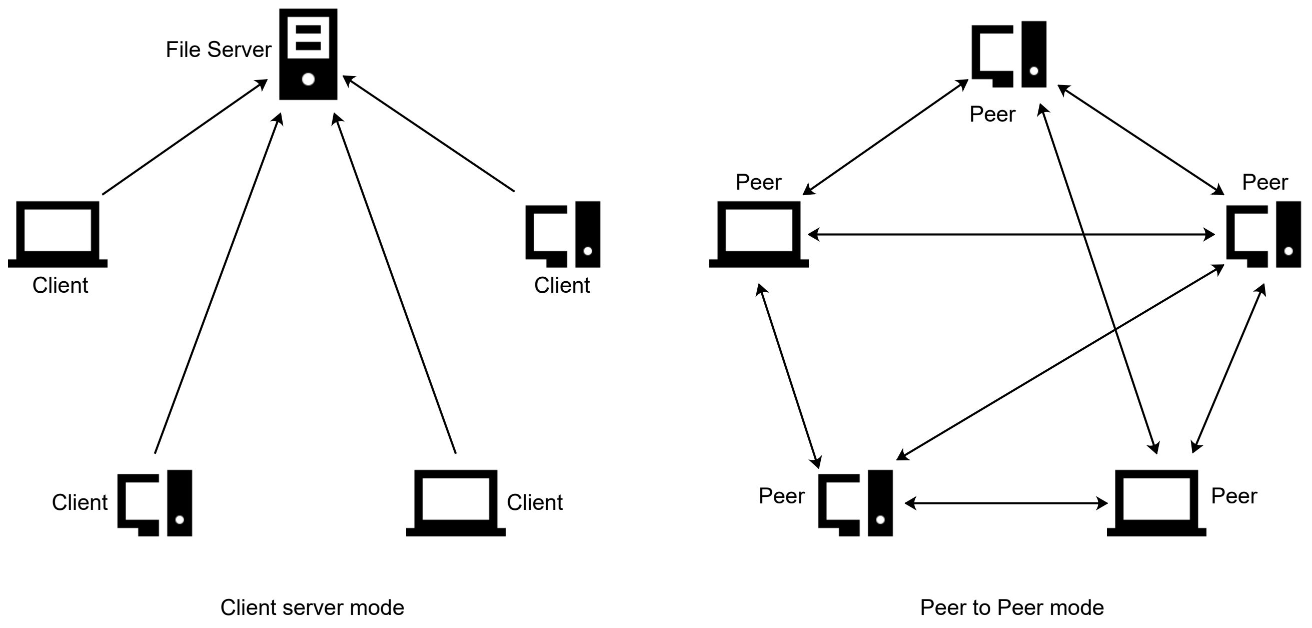


Figure 1.5 : Client server vs P2P mode.

4.3. Role of p2p in blockchains

Satoshi Nakamoto described it as a "***Shared Electronic Cash System***"[2]. In the beginning phases of Bitcoin[2]. It can be moved from one client to another through a P2P network, which deals with a distributed ledger called blockchain. In this specific situation, the P2P structure that is implicit to blockchain technology is the thing that permits the world wide exchange of Bitcoin and different cryptocurrencies, without the requirement for mediators nor any central server.

5. Blockchain technology

5.1. Definition

The core thoughts behind blockchain technology emerged in ***1991*** when assigned chain of data was utilized as an electronic ledger for digitally signing documents in a manner that could show none of the signed documents in the collection had been changed without any problem [1]. It was first applied to digital cash in 2008 in the initial paper depicting the Bitcoin electronic cash solution, Bitcoin: A Peer-to-Peer Electronic cash system [2], which was distributed pseudonymously by ***Satoshi Nakamoto***. The genuine author(s) and proprietor of the first Bitcoins stay a secret. From that point forward, blockchain technology has gotten closely connected to Bitcoin and it's regularly thought to be utilized for financial exchanges (despite the fact that it isn't limited to basic fund transfers). Nakamoto's paper contained the blueprint that most present day advanced digital cash schemes follow, with numerous varieties. Bitcoin is actually the first of numerous applications or use cases for a blockchain.

5.2. Blockchain structure

Each blockchain is organized slightly differently. However, the data is organized with the goal that each full node (the computers running the network) holds the entire data on the system. This model is persuading from the perspective of data persistence. It ensures that the data will stay intact regardless if some of the node are compromised. The way Blockchain arranges the organization and entry of new data incorporates three principle components[17] :

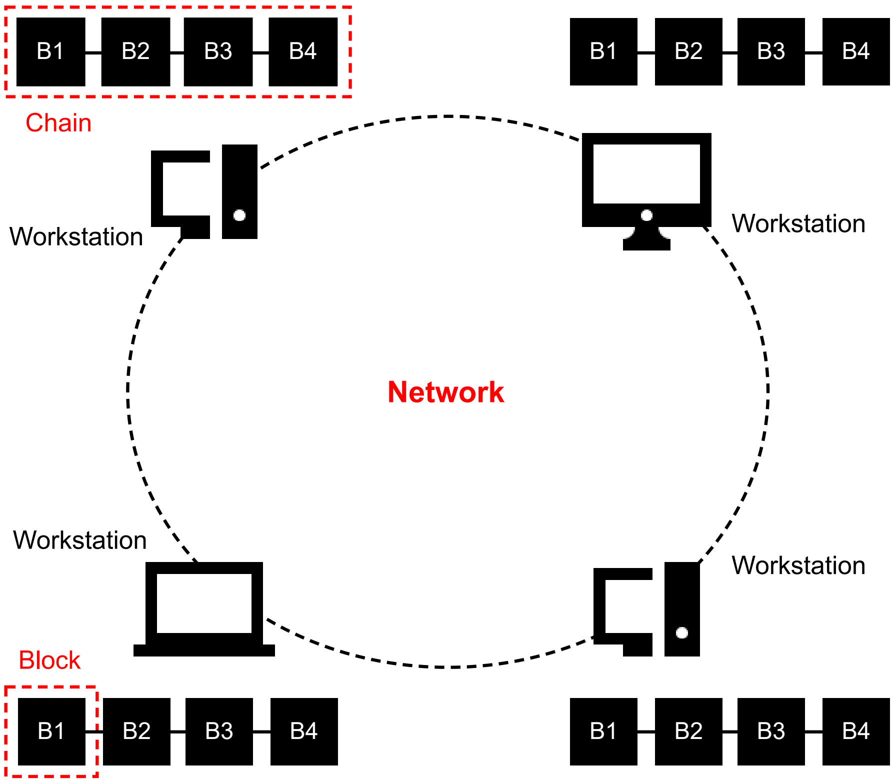


Figure 1.6 : Blockchain structure.

5.2.1. Network

The network is composed of "***full nodes***". Consider them as the computer running an algorithm that protects the network. Every node contains a full record of all transactions that have been recorded in this blockchain. The nodes are found worldwide and can be worked by anybody. It is troublesome, costly and time consumption to work a full node, so individuals don't do it for nothing. They have a motivating force to exploit a node since they need to acquire crypto currency [17]. The underlying blockchain algorithm rewards them for their service. The prize is generally a token or cryptocurrency.

5.2.2. Chain

Blocks are chained together through each block containing the hash condensation of the past block's header, in this way framing the blockchain. In the event that a formerly distributed block were changed, it would have an alternate hash. This in turn would make every single ensuing block likewise have various hashes since they incorporate the hash of the past block. This makes it conceivable to handily recognize and dismiss modified blocks[17].

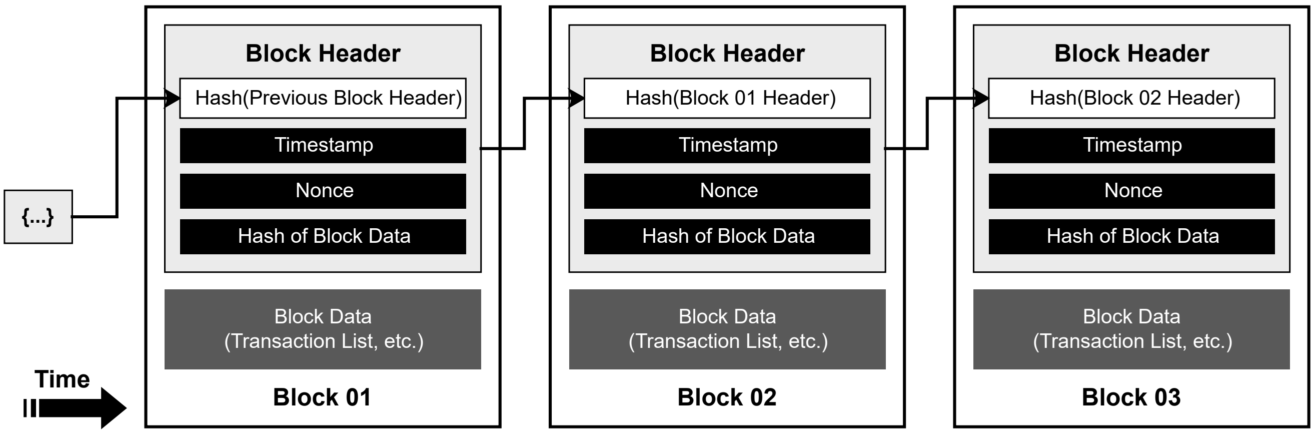


Figure 1.7 : Generic chain of blocks.[36]

5.2.3. Block

A block is a data structure in which advanced information is put away that is shared through a chain of blocks. They are sorted out linearly, and new transactions are continually prepared by miners new blocks that are added to the end of the chain [3], ***Table 1.1*** shows the structure of the block, it additionally contains information about the size of the block, records that show the content of the data in that block and their counter. There is likewise a header in the structure of the block, which will be clarified in more detail in the following section, and it contains metadata.

Name	**Description**	**Size**
Magic no	0xD9B4BEF9	4 bytes
Block size	Block size in bytes	4 bytes
Block header	Block metadata	80 bytes
Transaction counter	How many transactions does the block contain	1 – 9 bytes
Transactions	Transactions stored in a block	Variable

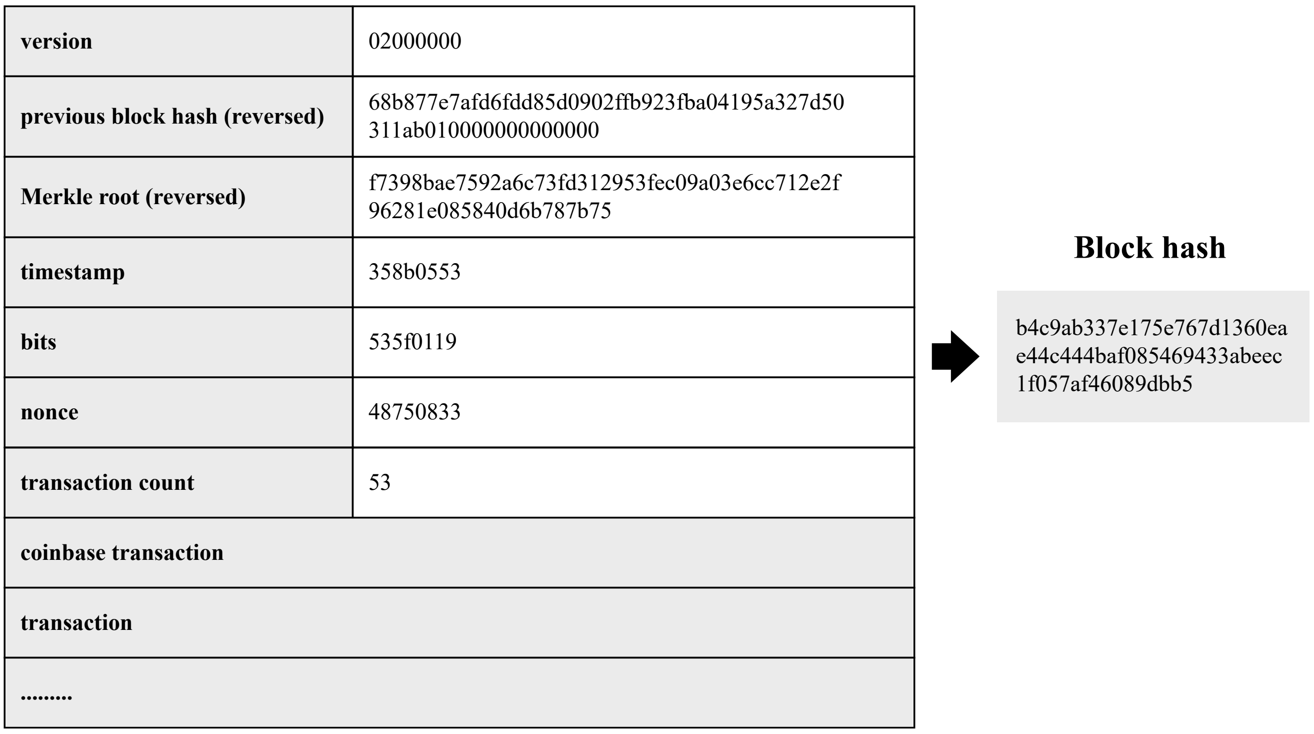
Table 1.1 : Block structure.[3]

5.2.3.1. Block header

To understand precisely what's happening to every individual block, it is important to see in more detail at the block header. The block header consists of **80 bytes** of String data that fill in as extra information about the block and the chain link. In ***Table 1.2*** made according to the source [4], the block header structure appears.

Name	**Description**	**Size**
Version	This field which provides the version number used in the block (Specific for Bitcoin).	**4** bytes
Previous Block Hash	The Block Hash of the block that this block is being built on top of. This is what "chains" the blocks together.	**32**bytes
Merkle Root	All of the transactions in this block, hashed together. Basically provides a single-line summary of all the transactions in this block.	**32**bytes
Timestamp	The timestamp begins when the miner started the hashing the header. This must be strictly greater than the median time of the previous 11 blocks.	**4** bytes
Difficulty Target	Also referred to as the Bits. The Bits specify a value or target threshold that contains leading zeroes. This is the basis of the difficulty target, which is not the same as the Bits.	**4** bytes
Nonce	The field that miners change in order to try and get a hash of the block header (Block Hash) that is below the Target.	**4** bytes

Table 1.2 : Block header structure.[4]



version	02000000
previous block hash (reversed)	68b877e7afd6fdd85d0902ffb923fba04195a327d50 311ab010000000000000
Merkle root (reversed)	f7398bae7592a6c73fd312953fec09a03e6cc712e2f 96281e085840d6b787b75
timestamp	358b0553
bits	535f0119
nonce	48750833
transaction count	53
coinbase transaction	
transaction	
....	

Block hash

b4c9ab337e175e767d1360ea e44c444baf085469433abeec 1f057af46089dbb5

Figure 1.8 : Example of block data.[4]

5.2.3.2. Hash functions

The hash function contention is data of arbitrary length, however the outcome is fixed. The entire blockchain technology depends on abusing the properties of hashes. The hash of a block is extremely simple to calculate, yet it is troublesome, it isn't even conceivable to discover what information is covered up out of sight of the determined hash. It is sufficient to transform one letter with some info data or sentence, the hash of that data will look totally changed. To demonstrate the above cases, we will list a couple of messages and their output value of the hash function ***SHA-256***.[5]

- **SHA-256**

  Cryptographic hash functions are mathematical operations performed on digital data. ***SHA-256*** is one of the ***SHA-2*** cryptographic functions whose standards were developed by the NSA (National Security Agency). The number 256 in the function name indicates the number of bits, while SHA stands for Secure hash Algorithm. ***SHA-256*** can be generated from any input or message, but the message cannot not be generated from a hash.[5]

  Sha-256(" *Akli Mohand Oulhadj Bouira University* **")**

  ```
  f7398bae7592a6c73fd312953fec09a03e6cc712e2f96281e085840d6b787b75
  ```

  Sha-256(" *Mohammed Merabet Mohamed Amine* **")**

  ```
  68b877e7afd6fdd85d0902ffb923fba04195a327d50311abf3056c31f5b899c0
  ```

  Sha-256(" *Boudissa* **")**

  ```
  b4c9ab337e175e767d1360eae44c444baf085469433abeec1f057af46089dbb5
  ```

5.2.3.3. Transactions

Transactions inside a blockchain network are data frame which store data among various locations for the exchange of values. On account of cryptocurrencies, the amount of currency that the sender sends to the beneficiary is the value transmitted through transactions. All transactions are open to the public and can be seen utilizing a self-named ***block explorer***. The block explorer really decrypts network transactions records into a comprehensible record [6].

- **Transaction verification**

  The transaction is checked before added to the transaction pool by a customer, and adheres to specific rules. In the event that the transaction doesn't adhere to the rules, it is marked as invalid by the node and inserted it into the transaction pool, with a few explicit rules. On the off chance that somebody makes a block with an invalid transaction, that is, breaches one of the revoked protocol, at that point the block likewise gets invalid and consequently different members in the chain's network containing such a block won't accept it. [6]

- ## Transaction pool

A transaction pool is a random access memory (***RAM***) space that is distributed to a mining device to store transactions that a node receives from the network. Legitimate transactions are placed first transaction pool. Miners at that point gather transactions from this transaction pool, and stop them in blocks. ***"Unverified transactions"***, that is, those that are not incorporated into any verified block, stay in the transaction pool and are accessible for expansion to the following block. Least transaction cost, transactions that don't have enough recorded costs are commonly remembered for the transaction pool and are overlooked. There is likewise a situation where not all the unused outputs listed in the transaction inputs were found to be the goat of the transaction [6]. A transaction of this type is considered an orphan transaction. Certain implementations of nodes additionally have a different pool for orphan transactions.

- ## Transaction structure

Transactions must be structured in standardized way with the goal that they can be read and comprehended by various devices and programs. The list of fields and their purposes can be found in ***Table 1.3***.

Field	**Purpose**
TxId	Transaction identifier
Version	Version of transactions. The first version of transactions has a field set to 1, transactions of subsequent versions have a larger number.
Size	Transaction size in bytes.
Locktime	The field in which we write the sequence number of the earliest block in which we want the transaction to be written.
Wine	A group of fields concerning inputs.
Vout	A group of fields related to outputs.

Table 1.3 : Transaction structure.[7]

5.3. Blockchain types

In light of the way that blockchain has advanced in the course of the most recent couple of years, it very well may be partitioned into numerous classes with unmistakable however in some cases somewhat covering qualities.

5.3.1. Public blockchains

5.3.1.1. Definition

Public blockchains are not anyone's. They are accessible to the general population and everyone can take part as a node in the decision-making process. Users may perhaps be remunerated for their support. All users of these ***permission-less*** or ***un-permissioned*** ledgers keep a copy of the ledger on their nearby nodes and use a distributed consensus mechanism to pick the last condition of the ledger [17]. The ***Bitcoin*** and ***Ethereum*** blockchain uses this mechanism.

5.3.1.2. Consensus (Proof-based)

Leader is randomly chosen (using an algorithm) and gives a final value using Roof-based, leader-election lottery, or consensus of Nakamoto. This group is often termed a form of decentralized or permission-less sort of consensus mechanism. Coming up next isn't an exhaustive list, yet it incorporates some outstanding algorithm.

- **Proof-of-Work (PoW)**

  Proof-of-Work uses the riddle solving method to demonstrate consensus block participation. In the event that a node wishes to take a part in the consensus block proposition, it must discover an answer before knowing any other blocks proposed for that given index. The participation determination admit any number of nodes as long as they have given a right solution. To create a legitimate block with Proof-of-Work [7], as shown in ***Figure 1.9***, a miner over and over submits a picked nonce to a hashing function alongside the predecessor block hash the root hash of the transaction and other metadata.

  To decide the legitimacy of the created output it's compared with the difficulty threshold. Based on the blockchain, the difficulty threshold, is legitimately corresponding to the amount of hashes to be done in order to discover a legitimate solution and function as a delay between proposals.

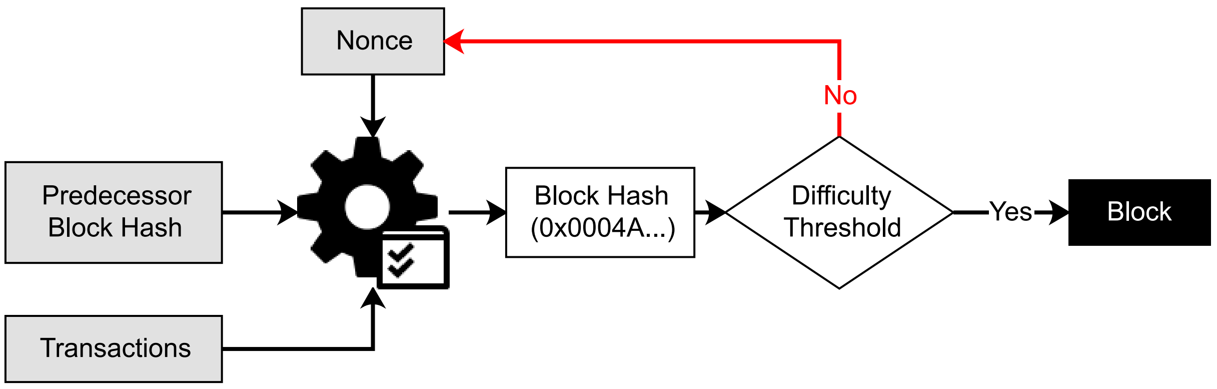


Figure 1.9 : Proof-of-Work flow.[7]

Objectives and assumptions

Proof-of-Work aims to give an open, permission-less environment membership determination, in which all nodes may join or take an interest openly. The likelihood of membership determination is related to the node's hash power, meaning an improvement in hash power requires an increment in the expense. Proof-of-Work computational adversary premises support this, that most of the network's computational power is held by legitimate nodes, where the likelihood of choosing a adversary on different occasions in a row is low. [7] Proof-of-Work is regularly joined by a prize system that rewards the nodes for their efforts in order to encourage nodes.

Weaknesses

Though Proof-of-Work accomplishes its objective of selecting blockchain membership, a various number of limitations hinder. The mining process required to solve the Proof-of-Work riddle requires a lot of computation, bringing about high asset and energy utilization. [7] The rising trouble to keep up a steady block proposition time brings about expanding costs for running a miner, vigorously affecting the scalability and future activity of the blockchain.

Proof-of-Work variants (CPU-bound PoW)

The miner batches a number of transactions to create a solution for the CPU-bound Proof-of-work and makes a Merkle tree [15] from the transaction hashes. The miner knows the worldwide edge, or trouble, the most recent block, and the transactions. At that point choose a nonce and apply a pseudo-random function to the new transactions block. The higher the complexity, the greater the leading zero is required in the hash estimation of a valid block. ***Table 1.4*** shows the connection between complexity, target, and the number of predicted hashes. The target is the amount that the block hash must be lower than, and as seen, with that complexity the number of leading zeroes increments, which takes more anticipated effort to solve.

Difficulty	Expected number of hashes	Approximate target
1	4.295e+09	0x00000000FFFF00000000000000000000...
100	4.295e+11	0x00000000028F5999999999A000000000...
1000	4.295e+12	0x00000000004188F5C28F5C2800000000...
10000	4.295e+13	0x0000000000068DB22D0E560400000000...
1000000	4.295e+15	0x000000000000010C6E6D9BE4CD700000...
100000000	4.295e+17	0x000000000000002AF2F2D14354120000...
1000000000000	4.295e+21	0x00000000000000000119787E99468E30...

Table 1.4 : The relation between difficulty and the expected number of hashes.

Memory-bound PoW

The memory-bound Proof-of-Work, otherwise called egalitarian Proof-of-Work, depends on slow memory access instead of computational hashing power, emphasizing latency on dependence. This renders the performance limited by speed of memory access instead of hashing power. Thusly, this gives the system resilience to ***ASIC*** miners and quick memory-on-chip devices.

- **Proof-of-Stake (PoS)**

Proof-of-Stake idea in blockchain was first introduced in 2011 in a Bitcoin community forum [8] to give faster and more definitive affirmation of transaction through a virtual mining mechanism. The fundamental thought of Proof-of-Stake is that the consensus members will deposit something of significant worth at stake, and this deposit will be removed if the node is seen behaving inappropriately. The virtual mining not just offers usage environmentally friendly blockchain network, by reducing total energy utilization compared with PoW, yet in addition improves the performance as blocks can be added to the chain more easily. The essential inspiration behind Proof-of-Stake is to reduce the energy loss brought by Proof-of-Work block creation process.[7]

The center idea lies in a node demonstrating legitimacy through staking resources, replacing the hashing to resolve a cryptographic riddle with a stake-based determination, while as yet retaining the permission-less idea of the blockchain. The stake is either taken from their existing balance, or the investor locks or deposits the stake. The node's voting power can be relatively planned to the stake they have given. ***Figure 1.10*** displays the Proof-of-Stake calculations.

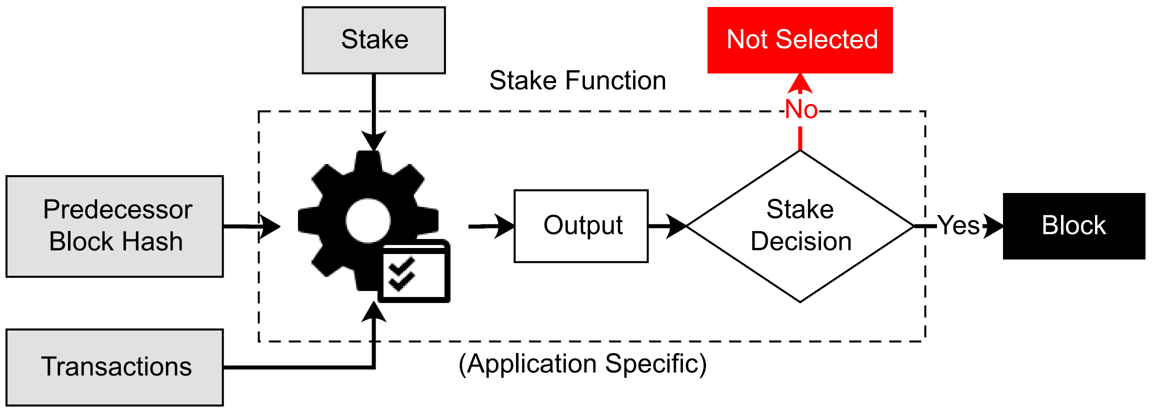


Figure 1.10 : Proof-of-Stake flow.[7]

Objectives and assumptions

All implementations of the Proof-of-Stake share some shared objectives and suppositions. Members are chosen utilizing balance, deposits, or votes, through stake estimations. Nodes obtain, or deposits more stake to increase the chance of selection. In such cases, for example, Delegated Proof-of-Stake [10], the nodes are chosen by voting, where the votes are weighted proportionally. Proof-of-Stake accept that there assets for a node to a stake, has a limited amount of resources and picking up the resources requires time or huge expense.

The different implementations give motivations by rewarding, or penalizing, actions, and cooperation. The chosen nodes that are frequently rewarded through transaction charges or block rewards. In any case, Proof-of-Stake is regularly joined by discipline; when a node is found sinful of trouble making, it loses the stake that it has deposits or owns. Like Proof-of-Work, Proof-of-Stake claims that an enemy doesn't control enormous parts of the stake, that the stake follows a distribution that takes into consideration that allows higher likelihood of choosing fair nodes than the adversary.

Weaknesses

Although Proof-of-Stake mitigates the use of resources linked to Proof-of-Work, it is range of numbers of inherent vulnerabilities are likely to occur. The unsolved problem with Proof-of-Stake is the expanded capability of centralization and the governance issues.

The distribution of assets, just as the willingness of people to stake their assets, extraordinarily impacts the membership determination and could result a little subset of nodes being in a place of intensity for the operation of the whole system. Many Proof-of-Stake implementations use periodic membership changes to take care of this issue, and incorporate randomness so the council changes, and a bigger pool of nodes is utilized. In any case, the use of incentive mechanism still incredibly affected that [9].

• Other protocols

Delegated Proof of Stake (DPoS)

It is an advancement compared to regular PoS, where every node that has an benefit in the system can delegate the legitimacy of a transaction to different nodes by voting. It is utilized in ***BitShares*** blockchain.

Proof of Deposit (PoD)

For this case, nodes wishing to participate in the network and before they can mine and propose blocks, nodes need to make security deposit. This process is used in the ***Tendermint*** blockchain.

Proof of Storage (PoS)

This structure takes power limit redistributing into consideration. This structure relies that most likely a particular bit of information is deposits by a node which fills in as a way to the consensus mechanism.

5.3.2. Private blockchains

5.3.2.1. Definition

A ***permissioned*** blockchain is a private blockchain. Private blockchains operate on getting controls which limit the members who can take part in the network. There are at least one substances that manage the network and this contributes to third-party dependence on transactions. Only the members involved in transaction will know of this transaction in private blockchain while other others won't have the option to get to it. Linux Foundation ***Hyperledger Fabric*** of is an ideal case of a private blockchain. [17]

5.3.2.2. Consensus (BFT-based)

BFT-based is an increasingly conventional methodology dependent on rounds of votes. This form of consensus is otherwise called the consortium or permissioned type of consensus mechanism.

- **The byzantine generals problem**

  The Byzantine Generals Problem describes a circumstance in which the Byzantine army is divided into various units, each directed by a general. The army decides to make an assault on the city, but the assault can succeed if the units are simultaneously prepared to attack. The generals must communicate through messengers to prepare a planned attack and make decision whether to attack or retreat. But, one or even more generals might be betrayers who may attempt to try to sabotage the attack so the faithful generals can't achieve a coherent plan. The objective is to get each truthful general to agree to a similar choice even within the sight of betrayers [14].

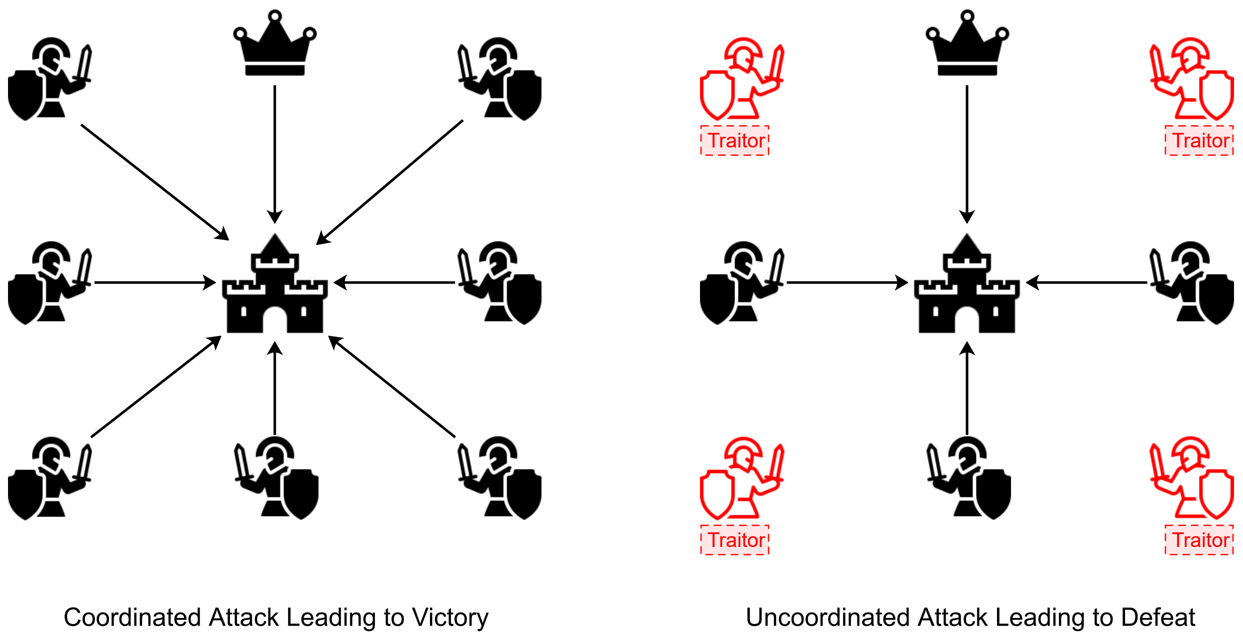

Traitor
Traitor
Traitor
Traitor
Coordinated Attack Leading to Victory
Uncoordinated Attack Leading to Defeat

Figure 1.11 : The byzantine generals problem.

To represent the issue, let the army have ***2N+1*** generals altogether, and one of them is a betrayer. When half of the faithful generals need to assault, and half needs to withdraw, the betrayers can misguide the decision of generals to assault or retreat.

This can prompt a circumstance where ***N*** generals assault and ***N*** retreat, lethal for the Byzantine armed force. This issue get much worse as emissaries are utilized to communicate and the traitor may keep messages from being sent to a trustworthy general, or by writing a false message.

- **BFT Protocols**

Standard BFT protocols offer answers for this issue through procedures, for example, the utilization of signatures, having the option to identifying the absence of messages, or something else. Considering the setting of the blockchain, where various nodes might have the ability to act maliciously or encounter irregular network traffic, BFT protocols have been an acceptable determination to achieve consensus.

The Practical Byzantine Fault Tolerance (***PBFT***) [11] algorithm accomplishes consensus via leader-based correspondence by three steps. The primary step, pre-prepare, starts with the leader giving a sequence number and multicasting to every single other node. When a node gets ***pre-prepare*** ready, it enter the get prepare step and multicasts to every other node. In the event that a node gets ***2f*** prepare messages matching the pre-prepare message, where ***f*** is the quantity of possibly defective nodes, it enters the submit stage and multicasts a submit message. If a node had gotten sufficiently commit messages from ***2f+1*** copies that suit the pre-prepare message is then committed. The pre-prepare and prepare stage are utilized to organize the messages, while the planning and commit stages are utilized to guarantee the commit requests are completely organized over all nodes.

For the situation that the primary shows flawed behavior, or the reproduction "timers" expires, PBFT provides view-change mechanism that chooses another primary. Both replicas must send the VIEW-CHANGE message to all replicas containing the current state to effect the view-change. The new primary at that point sends the NEW-VIEW message to all reproductions to activate the new view.

Systems such as *PeerCensus* [12], *ByzCoin* [13], and *Solida* [15] use variations of the PBFT algorithm customized to their requirements to provide BFT to the blockchain.

Objectives and assumptions

PBFT variants discussed in this study, *PeerCensus* [12] and *ByzCoin* [13], show comparative objectives and hypotheses. Consensus can be achieved in a partially synchronous domain, where Byzantine behavior exhibits strictly below of the nodes. *Solida* [15], though accept a sequential domain to guarantee that a delta message delay is known from the earlier. Nevertheless, *ByzCoin* offers a version that uses two-stage signature scheme to achieve an agreement by using signed messages in leader-election consensus protocol. The ending that of PBFT, is accomplished when the nodes vote on the block presented.

5.4. Blockchain applications

The way transactions are being done and constantly forming it and could turn into a vital piece of lives because of the enormous effect has Blockchain as a technology. In different businesses there are such a large number of utilizations of blockchain technology, which incorporates however isn't restricted to the few of mentioned beneath.

5.4.1. Patient records

Blockchain provides a chance for interoperability in healthcare service as having as providing a shared ledger of clinical records where all healthcare services suppliers have access to this decentralized ledger. Although the UIs might be unique, this ensures that their central ledger will be the same for all suppliers. A problem that occurs relates with the present status of health records across suppliers that contain large quantities of a similar data under various identifiers that may not be connected. [17] It causes replication and as the blockchain expand, the performance debases and this degree of replication of information across records would expect reduplication to keep up a fairly functioning system with specific, anonymized identifier to distinguish patients over all services.

5.4.2. Drug tracking

Drug tracking on the blockchain is another open door as it uses the changelessness of the blockchain to establish monitoring and chain of guardianship from supplier to consumer. This helps healthcare suppliers to fulfil current healthcare insurance guidelines in regards to protection of pharmaceutical supply, again with an accentuation on interoperability between healthcare suppliers.[17]

The source of counterfeit pharmaceutical goods can be tracked and withdraw from the ***supply chain*** using Blockchain.

The benefit of blockchain drug following over conventional methods is the decentralization of confidence and authority inherent in the concepts underlying the technology, where central authorities can be paid off or fooled it is a lot harder to pay off an agreement of those on the blockchain.

5.4.3. Voting

Voting fraud accusations have just happened as lately as the last U.S presidential election. Considering the of use PC system which sometimes, cost a large number of dollars, fraudsters are finding more and more creative opportunities to manipulate them.[17]

Smart contracts are a straightforward and affordable answer for this issue. They can be utilized to approve a voter's identity and record their vote. Once the voting had stopped this data could then be utilized to start an operation. Since the blocks inside a blockchain can not to be change once they have been registered, it would not be possible to manipulate this record.

Conclusion

During this chapter, we mentioned some information about decentralization and distributed systems and the difference between them and peer-to-peer network, which was the foundation for the creation of blockchain technology, after that we did a deep analysis about this technology and explained two types of it with the important part the consensus mechanism in it, with mentioning several examples (applications) used with this technology, as we discussed the structure of block to find out all its components and the role of each one of them, and as a result of this research we think that the blockchain despite its progress, it gives big hopes with the exploitation of all its advantages granted to create safe environments and more responsive to sabotage.

Chapter II

Smart Contracts, Supply chain and Internet of things.

Introduction

In this chapter, we will dive deeper into blockchain technology, especially in smart contract technology, its types and applications, as a complete to the previous chapter. Then we proceed to the second section of this chapter where we will present two types of technology on which we will construct our final project, namely supply chain and IoT. We mentioned the barriers, limitations and applications for each.

1. Smart contracts

1.1. Definition

A smart contract is an executable code running on the blockchain to promote, execute and implement the conditions of a transaction. Smart contract fundamental function is to execute the term of agreement automatically until the conditions stated are met. In this way, smart contracts guarantee low transaction charges relative with conventional systems that require a confided in outsider to authorize and execute the terms of an agreement. In 1994 ***Nick Szabo*** came up with the concept of smart contracts [20]. The thought wasn't seeing the light till the rise of blockchain technology appeared. A smart contract can be considered as a system that discharges resources for all or a portion of the included partied once subjective pre-characterized rules have been met [21]. For example, if Alice receives ***X*** currency units from Carl then she sends ***Y*** currency units to Bob.

The literature has addressed many different concepts of a smart contract. In [22], the creator grouped all definitions into two group, to be specific, smart contract code and smart legal contract. Smart contract code signifies "***code that is put away, confirmed and executed on a blockchain***" [22]. This smart contract functionality relies altogether upon the programming language used to communicate the contract and the blockchain's features. Smart legal contract implies code to finish or changing legitimate contracts. This smart contract's capacity doesn't rely upon the technology, however on legitimate, political and business organizations instead.

There is a balance of account, private stock, and executable code in smart contract. The state of contract involves the storage and the contract balance. The state is put away on the blockchain and it is refreshed every time the contract is invoked. The smart contract structure is shown in ***Figure 2.1***.

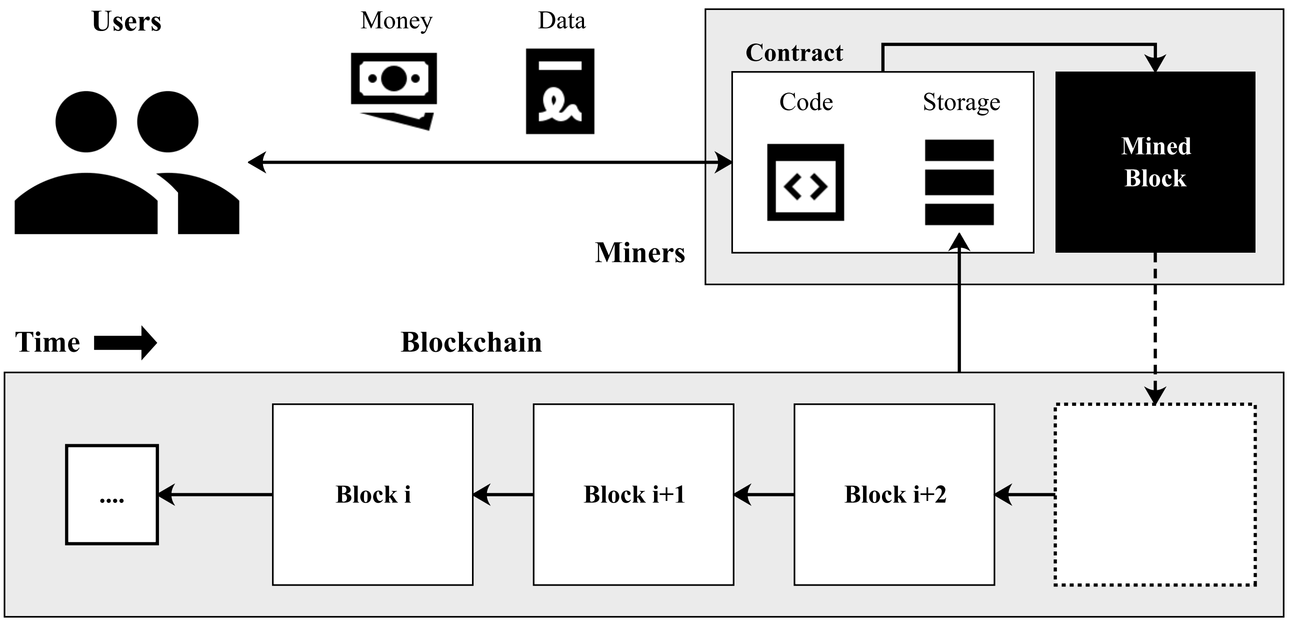


Figure 2.1 : Smart contract system [24].

Each contract is allotted to a one of a kind 20 bytes address. When the contract is implemented in the blockchain, no modification can be made to the contract code. Clients will essentially send a transaction to the address of the contracts to run it. At that point each consensus node (***miners***) in the system will execute this transaction to arrive at an accord on its output. The condition of contract will at that point be refreshed accordingly. The contract will read/write to its private storage, store money into its account balance, send/get messages or money from clients/different contracts, or even make new contracts, depending on the transaction it gets.

There are two sorts of smart contracts, in particular smart contracts that are ***deterministic*** and ***non-deterministic*** [23]. A deterministic smart contract is a smart contract which it doesn't require any input from an outer party (from outside the blockchain). A non-deterministic smart contract is a contract that relies on input from an outer party (called oracles or data feeds). For instance, a contract that require the current climate data to be run, which isn't accessible on the blockchain.

1.2. Smart contracts functionality

Basically, smart contracts operate just like sales machines. Only drop a necessary measure of cryptocurrency into the smart contract, and your drop it into your account with escrow, house possession right, driver's license, or whatever else [23]. Not only are all laws and punishments decided by smart contracts but they are also authorized by them.

1.2.1. Interdependence

A smart contract can function on its own, yet it can likewise be applied in combination alongside any number of other smart contracts. They can be set up in a manner when they are going to rely on each other. Effective completion of one specific smart contract, for example will cause the beginning of another. In principle, they can operate entire system and associations completely on smart contracts. [23] Somewhat, it's already applied in different cryptocurrency network, where all the laws are defined and thus the network itself can work independently and autonomously.

1.2.2. Objects of smart contracts

Basically, every smart contract contains three key component, referred to as items. The first is ***signatories***, the two or more parties utilizing ***the smart contract***, accept or contradicting the terms of the contract utilizing digital marks.

The second item is ***agreement*** topic. It can only be an entity that exists inside the context of the smart contracts. Alternatively, the smart contracts have to control the object unhindered and directly. Although the smart contracts were first debated back in 1996, their creation was delayed by this particular item. This problem was partly solved only after the introduction of first cryptocurrency in 2009.[23]

At last, any smart contract needs to contain ***unique*** terms. Those terms must be defined in fully mathematically and must utilize a programming language that is suitable for the specific smart contracts. This includes the necessary criteria of all the involved parties just as all the laws, prizes and disciplines related with those terms.

1.2.3. Environment

With the end goal for smart contracts to exist and work appropriately, they need to work inside a particular reasonable condition. First, the system needs to promote the utilization of ***public-key*** cryptography, enabling clients to sign off for the transaction utilizing their special, specifically created cryptographic codes. That is the exact method which is used by larger part of existing cryptocurrencies. [23]

Also, they include an open and decentralized database that can be completely trusted by all parties to the agreement and which are completely automated. Additionally, the smart contract to be applied, the whole system must be decentralized. Blockchains are the ideal environment for smart contracts, particularly the Ethereum Blockchain.

At last, the digital information source utilized in smart contract must be totally accurate. This involves the utilization of root ***SSL*** security certificates, ***HTTPS***, and other secure-connection protocols which are already commonly utilized and are automatically implemented on most modern software.

1.3. Smart contracts types

Smart contracts adopt the properties of hidden blockchains that incorporate a changeless data record, and the ability to mitigate single purposes of disappointment. Also, smart contracts can communicate with one another through calls. In contrast to conventional paper contracts that depend on mediators and outsider mediators for execution, smart contracts simplify contractual method, limit conflicts among parties, and reduce organization cost. Smart contracts on public blockchains or "public smart contracts have pulled in a wide assortment of business applications because of the simplicity of arrangement. [25] While smart contracts on private blockchains or "***permissioned smart contracts***" are more frequently utilized in collective business forms since they can possibly avoid undesirable updates, improve effectiveness and save expenses.

/	**Public Smart Contracts**	**Private Smart Contracts**
Common	a. Immutable record b. Proper encyption on data and pseudonymity c. Interoperability among different platforms d. Traceable modifications	
Unique	a. Easy to deploy b. Accessible for the public	a. Faster settlement b. Lower operational cost c. Permissioned access

Table 2.1 : Characteristics of public and permissioned smart contracts.[25]

There are two sorts of smart contracts, to be specific smart contracts that are deterministic and non-deterministic [23]. A deterministic smart contract is a smart contract which doesn't require any input from an outside party (from outside the blockchain) when it's run. A non-deterministic smart contract is a contract that relies upon data from an outside party (called oracles or information feeds). For instance, a contract requiring the current climate data, which isn't accessible on the blockchain.

They divide smart contracts into public and private and discuss the legal nature of smart contracts and their usability. Elements encapsulate information to be obtained and visualized or implemented, and/or communicate with application logic. What the element returns is a feature of the kind of the element.

For smart contracts we can recognize the following contract types :

1.3.1. Generic contracts

Conventional contracts incorporate application logic, e.g, for management of deposits which can be summoned by blockchain customers or by different contracts, this form of contract is commonly specified in that it retains application state across interactions.[26]

1.3.2. Libraries

Execute at least one operations, e.g, a math library, that are intended to be reused by different contracts[26], libraries don't store inside and are stateless internal variables.

1.3.3. Data contracts

Offer information storage services inside the blockchain, e.g., a manager of consumer references intended for use by different contracts[26], they are stateful by nature.

1.3.4. Oracles

Deliver information services from the outside of the blockchain to within blockchain, e.g., rates of conversation with currency. Contracts can't make calls outside the blockchain, as external dependencies will forestall verification (transformation levels change after some time). If information is required from the outside, customers may send it to oracles utilizing transactions[26], this would then permit different contracts to request the information.

1.4. Smart contracts efficiency

We can make exceptional improvements by applying smart contracts in our everyday life, as they offer numerous points of advantages over the traditional contracts. Smart contracts are simpler and more convenient, making them appropriate for individuals to streamline out their work processes.

They give you the correct mix of protection and application simplicity when you have to trade anything of significant value whether it is land, cash or shared.

Taking out the requirement for mediators make for an even more exciting the use of smart contracts in our lives. The use of smart contracts is probably going to equip with the technological of development. Let us take a peek at the advantages smart contracts offer [34]:

1.4.1. Transparency

Transparency is one of the fundamental features of blockchain technology which is additionally shared by smart contracts. As recently mentioned, smart contracts are loaded up with terms and conditions which are likewise reviewed by the parties engaged in the agreement.

This reduces the opportunity of later stages conflicts and issues as the terms and conditions are carefully reviewed and registered and placed in place only when all the members agree. This smart contracts feature helps the parties included to guarantee transparency during transactions.[34]

In addition, the requirement for accuracy in contract details leaves all the data available for anyone who eventually addresses everything relevant to the question of miscommunication. Hence productivity lost in contact gaps can be re-established with the help of smart contracts.

1.4.2. Time-efficient

It normally takes more than a couple of days to proceed with any procedure including paperwork. Often mediators and unnecessary measures along the way are responsible for the delay in any processes. At the other side, smart contracts are operated because they only bits of software code with the help of the internet.[34]

Hence, the pace at which smart codes complete transactions is way too quick. Compared with any conventional business method smart contracts can save hours or even days. In addition, also the time delay because of manual participation is removed.

1.4.3. Precision

Coding a smart contract is in an expressly structure manner. It needs to keep all the terms and conditions in it until it is ends up being put to use. Any condition that is kept separate from the contract may bring about a failure during execution, so all the conditions are placed in the comprehensive structure when making smart contracts.[34]

Because of this, the smart contract turns into a reliable agreement which gets nearly everything completes when implemented automatically. There are odds of mistake in manual contracts as the individual responsible for creating a contract may forget one condition or another. In fact, it’s absolutely impossible to even following it until the mistake is made. Smart contracts are also a stronger choice when it comes to accomplishing precision and accuracy.

1.4.4. Safety and efficiency

In the current occasions smart contracts with mechanized coding functions are the most secure choicer when it comes to information encryption technology. Since they follow the best quality requirements, the degree of security they provide permits them to be secure to use for sensitive procedures.[34]

Furthermore, because the smart contracts are so precise and stable, their efficiency level is much higher to produce more value in transactions.

1.4.5. Data storage

Smart contracts are precise and reliable to the accord’s minimum degree. All details of any transaction are put away on the contract and it can be accessed at some random time by all the parties included. In addition, such transactions are put away in the form of future records on the blockchain. This will be especially useful in any potential conflict over the terms of contract later.[34]

1.4.6. Savings

Instead of conventional agreements, utilizing smart contracts can bring a great savings. As a matter of first importance because smart contracts include parties that are the piece of the agreement; the requirement for mediators is removed and the money included in that is also spared.[34]

By utilizing smart contract, all the legal judges, witnesses, and mediators have no job. In addition, as already stated, smart contracts likewise save money as paper-based archives are not included in any procedures.

1.4.7. Trust

Transparency and security properties make smart contract dependable in organizations. They demolish any chance of manipulation just as manual mistakes and build up trust in their execution. The contract would immediately executes itself upon agreement on all of the terms.[34]

Another exciting element of these contracts might be their potential to substantially decrease the lawsuit of suit and courts. Self-executing Smart Contracts require gatherings to agree and connect in accordance with the conditions and rules composed inside.

1.5. Smart contract applications

1.5.1. Public smart contracts

Public blockchains make it possible to build and test smart contract applications or decentralized applications (***D-Apps***). Public smart contracts allow start-ups to fundraise through Initial Coin Offerings (***ICOs***). On the other hand, Enormous companies mostly need to take advantage of permissioned smart contracts to incorporate their models and implement business processes. A few of the cases of common include using:

Banking, Electronic Medical Record (***EMR***), IoT management [27], Smart waste disposal, as other fascinating applications, for example real estate, and ride-sharing arcade city.

We directed a complete survey of existing smart contract applications and talk about their strength, limitations, and larger adoption prospects.

- **Health care and medical records**

  One big smart contract technology zone relates to medicinal services and access control of medical records. Most healthcare experts see blockchain technology and smart contracts as a protected method of exchanging and getting EMR for patients. Smart contracts can include multi-signature authorizations among patients and suppliers so that approved clients or devices can only obtain or append the record. They additionally allow for interoperability to keep up the record continuity through shared version control. In addition to benefiting patients and their care providers, smart contracts may likewise be utilized to supply researchers with access to certain individual health information and allow automatic transfer of micro-payments to patients for participating [28].

  The knowledge of these applications in any case, is restricted by the weak foundation of most public blockchains and high costs of growth. There are additionally worries about strategies and the ability of clients to broadcast their own data.

- **Identity management**

  uPort is a framework for identity management that use public Ethereum smart contracts to recuperate accounts and secure client privacy on event of device loss. The uPort identifier key component is a one of a kind 20-byte hexadecimal string representing the address of a proxy contract that lies in the middle of a controller contract and an application contract. uPort allows clients to change their private key (spared off-chain) while retaining a persistent on chain identifier. When a legitimate client brings another device, a list of existing recuperation delegates will request permission and replace the old client address with another one. Likewise, ***Sovrin*** is a form for digital identity management based on a public blockchain.

  Identity management structures that utilize blockchain still need to experience various changes before they are adopted. On account of uPort, the advertisement of the client's restoration delegates presents the security danger of compromising client identities.

1.5.2. Private smart contracts

Public smart contracts threaten client privacy inevitably. Cases of more business use cases, for example, banking, supply chain, and IoT are all the more regularly sent as permissioned smart contracts. We give debate on a portion of these cases of use below.

- **Banking**

  For instance, the home loan service, smart contracts can be used to implement rules and policies in banking. As indicated by a study made by ***Capgemini Consulting*** [29], with smart contracts in the home loan, customers might spare 480-960 USD per loan, while banks would have the option to cut yearly expenses in the US and Europe 3-11 billion USD. Banks will likewise utilize smart contracts to smooth out processes for clearing and settling. More than 40 worldwide banks have been reportedly taken an interest in a consortium to test smart contracts for clearing and settlement contracts [30]. Additionally, knowing your client (KYC) and antimony laundering (AML) strategies can be integrated easily with the smart contract logic. Based on top of Hyperledger Fabric, Stellar Blockchain allows facilitates programmed currency trade in International transactions.

  In any case, in the realization of these framework, interoperability with existing systems and the scalability of blockchains remain barriers. It is also urgent that the implementation of smart contract is protected against assaults that are planned for taking of assets or interfering with contract code [31].

- **Voting**

  Voting is another application capable of benefitting from permissioned smart contracts. A Danish political party has adopted a smart contract to guarantee interior elections are fair and transparent [32]. ***Mccorry*** et al. [33] suggested a boardroom voting system that would vary from current e-voting proposals. Mccorry's framework works under the presumption of a select number of electors with defined characters, ensuring full anonymity and variability for the elector. Mccorry et al. have evaluated the viability of the program on an private Ethereum system and calculated the expense of 0.73 USD per voter for running it. The measurements have demonstrated that public blockchains are more doable for little polls, though permissioned blockchains are needed to run elections on national scale.

2. Supply Chain

2.1. Definition

Supply Chains can be found in almost every industry, in some way, spanning several operating zones. A supply chain typically envelops all the procedures and activities leading from the initial raw materials to the last completed item, just as all the operations and services inside and outside a business. A supply chain can likewise be characterized as the network of substances where the material flows through. Such organizations can be defined as providers, carriers, fabricating sites, distribution centers, retailers, and clients [44]. Normally, with the upstream and downstream progression of these materials and assets, a lot of knowledge comes about them and about the procedures, people and associations which they are related with. Realistically, as there are many considerations to be taken and choices to be made, the flow isn't generally arbore scent. Supply chains are equipped with different final results with shared components, facilities, and limits [45]. As a result, the ways taken by the assets and data are not direct, however interlacing, diverging and a converging at various focuses, going back and forth, as shown in ***Figure 2.2***.

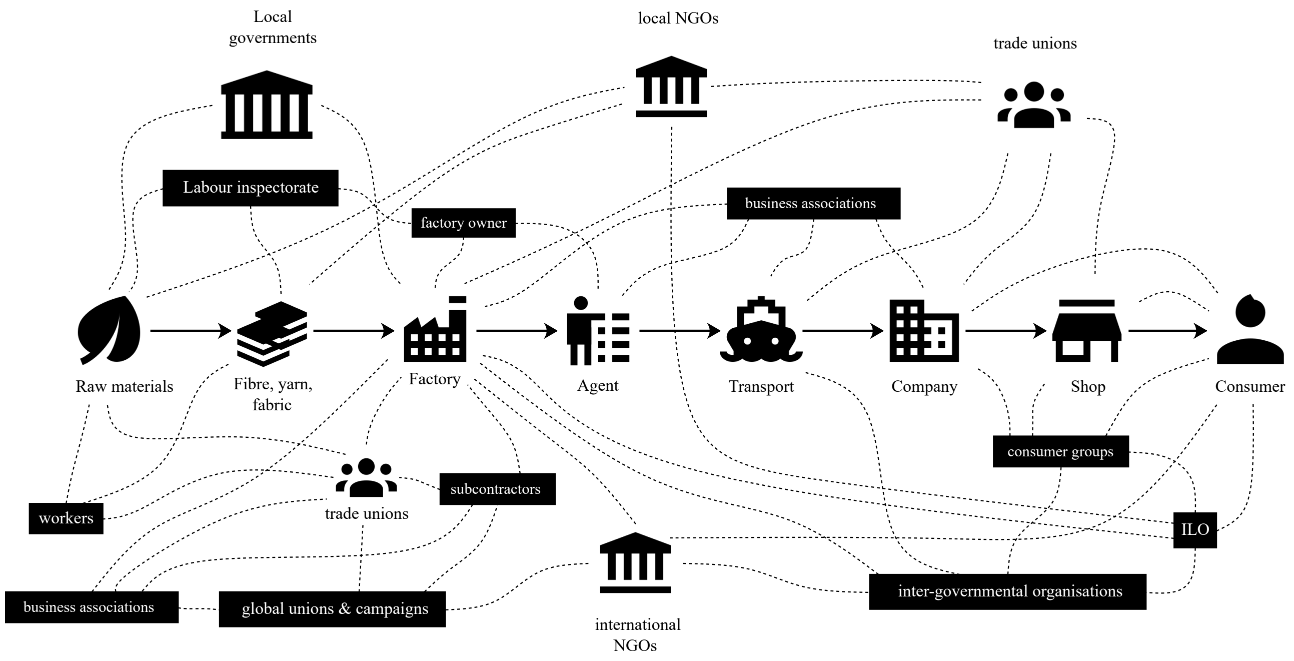


Figure 2.2 : Garment Supply Chain[43].

A supply chain's operations and procedures involve sourcing of raw materials and components production and assembly, warehousing and stock tracking, order entry and order management, distribution all over the network, delivery to the client and dealing with the data systems important to control all these operations. There operations, as defined by ***Lummus*** [44], can be roughly mapped to the 4 fundamental procedures: plan, source, create, and deliver.

It follows from this concept that ***SCM*** manages both coordination and cooperation among elements, it is very important to manage the flow of Knowledge and assets between them. Naturally, the goal is to always reduce the total expense of these flows between and among stages [46]. What's more, this is the place SCM shines and demonstrate exactly how helpful it may be. It is troublesome to deal with all the procedures in a supply chain, while preserving security, quality and keeping to a schedule. An occasion on one side of the planet, huge or little, whether caused by human or natural causes, may easily interrupt in the supply chain linkages.

SCM reduces the effect of these disturbances, and works aggressively to prevent or reduce them while improving the workings of the supply chain. That's the reason SCM is such a significant discipline that we need to better understand and improve, with all the means that we can, incorporating research into technologies like the blockchain of course.

2.2. Difference between supply chain and logistics

Logistics itself is usually widely applied to an individual company and the fundamental aim is to allow the best impact from the raw material to the end-clients on the logistics network of the company. SCM is wider than that, it fundamentally gowns from the logistics idea. But SCM is focused on outside collaboration with all the supply chain members. The goal of this partnership is to improve exchange the entire chain. Consequently, logistics became one piece of the supply chain and developed more efficiently through the chain.[35]

Applying the SCM involves enhancing the entire supply chain, while not relying exclusively on sub-optimization of just a single company. For instance, the producer must consider the cost of excess inventories to figure the last cost of any item. This is the link with delivering just-in-time. At the point when the manufacturing receives more benefit if they delay material procurement until the order has been set. As far as logistics is concerned, the plant just arranges how the material and item travel within an individual system.

With respect to SCM, it incorporates numerous players, with integrated main object. In any case, the collaboration among buyer and seller for the vast majority of the practical cases. Expanding to client or seller's merchant is rare. Any company, on the other hand, will see it from SCM point of view.

The explanation is the means by which well the connection and flow through the chain will choose directly how much benefit the supply chain brings and how much cost it'll save not just for one organization well as each organization in the chain. For instance, one producer needs to cooperate with numerous providers in the car business and every provider needs to cooperate with a various providers. So it shapes the cooperation of hierarchical vendors. In many chain cases, the jobs are not clear [35]. A car company can also be the provider for their rivals.

2.3. Supply chain challenges

Having just presented the ideas of Supply Chain and SCM, it is currently possible to quickly present some of the issues that affect them.[43]

A supply chain's first, and most generalist issue is the ease with which an unforeseen occasion may cause delays. These occasions, are not generally predictable and should be considered as quick as possible. Specifically one occasion that frequently causes delays is synchronization issues in the organization processes and data system [47].

Another challenge is that the exchange of knowledge between organizations also poses difficulties. It is caused by both the reality that organizations respect their privacy and the confidentiality of their data, implying that they may not have any desire to share too much amount of data, or that they can just share it through secure channels and by the absence of standards for sending data and transmitting information [48]. The problem with non-existent principles is that it is left to organizations to talk about what details to share or not, wasting time and assets.

Generally significant of all, the utilization of conventional methods and manual labor is much too prevalent in the industry. Emails are sent, reports are printed and sent, rather than the data being transmitted via the network in a more automated direct, and secure way. This point additionally brings the following issue of supply chains to light: the evident absence of interoperability between certain software (which may be a result of by the absence of standards).

Lastly, the provenance and detectability of the items on a supply chain are a major target for organizations. But existing technology utilized in supply chain just achieve provenance and traceability in a restricted setting, as the knowledge owned by a specific element is typically restricted as well. Thus, having a worldwide overview of the supply chain is really difficult for anybody.

In spite of the fact that it isn't demonstrated, it is be conceivable that a few, if not most, of these issues in supply chain may be triggered by the using software structures that don't enable complete information incorporation. An ideal supply chain should be as proficient and successful as could be expected under the circumstances while being secure and fulfilling all the traceability requirements. Maybe, the time has come to try new solutions that supplant or enhance the current ones, so that supply chain management can better meet the required necessities.[43]

2.4. Supply chain and Blockchain

Supply Chain Management (***SCM***) is one field in particular where we believe blockchain could add positive improvements. Because of the globalization of the market, SCM has seen an expansion in complexity in recent decades, with organizations intertwining from various perspectives, their relations expanding far past what they used to, as ***Filiz Isik*** found [49]. This rise in complexity is very difficult to handle because some supply chains reach and envelop such a large number of organizations that, since their software is not being set up for this, the data isn't constantly transmitted from end to end, leaving gaps of data between the connections that connect each company, leading to a lot of confusion and vulnerability about the condition of the main products in the chain [50].

2.4.1. Blockchain in supply chain Advantages

Blockchain would bring a portion of the benefits to supply chain, over different solutions: [50]

- **Less error-prone :** reduction of manual data entry errors, in particular when combined with IoT and other automated processes;
- **Enhanced security of transactions :** not only the ledger is permanent, fraud attempts are easily detected.
- **Improved tracking :**the ledger is easy to interpret and delivers results very fast, allowing the status of any request or resource to be known at any time; simultaneously, any mistake, either unintentional or intentionally, that figures out how to discover its way into the system is easily traceable.
- **Improved consumer trust :** blockchain will allow users to verify the provenance of their goods and establish a trust relationship with the suppliers.
- **Reduced costs :** reduced administrative costs for sharing data and so on, allowing greater productivity and quicker occasions for preparing the data (improving cost-effectiveness); decreased internal management costs, improved performance and competitiveness; decreased item or service costs, making strategic advantage and barriers to rivalry, reduced lead times in the supply chain and improved flexibility.
- **Internal supply chain trust :** It is significant that the supply chain trust the data that goes back and forth from one another, and blockchain permits this to happen.

Some of the most significant things is this last point, which is frequently ignored in favor of other more apparent functionalities.

Casually, a business realizes that at any time it can without much of a stretch access its own information so there is obviously a lot of self-confidence. Organizations in this manner trust and depend a great deal on themselves, however not generally on others. Trust implies being helpful and depending on different business for the required information, and accepting that this information error-free as well.

Cooperation and confidence as defined by ***Panayides*** [51], are the keys to improving efficiency and creativity of the supply chain, expanding the quality and bringing benefits for all parties concerned. ***Likewise***, ***Yeung*** [52] figured out how to discover a partnership among trust confidence a greater integration of the supply chain. The confidence could be described not only as loyalty as well as dependability of the supply chains. This last factor is very critical because it determines exactly the amount you can demand from your supply chain partners. What's more, confidence in the form of reliability is a significant advantage to have when you need fast details.

In this sense, if the blockchain technology figures out how to improve the data flow in a supply chain while preserving protection and confidence between parties (in any event at a technical level), at that point it follows that, just as Panayides and Yeung concluded, the supply chain itself would have an improvement in efficiency, as the parties included don't need to stress about these or any of aspects. Along these lines, confidence is by all accounts a key factor in creating a supply chain that is reliable and successful.

2.4.2. Challenges of blockchain application to supply chain

The opposite side of the coin is that Blockchain isn't generally as acceptable an answer as it is predicted. Blockchain itself is research subject and, although a portion of its applications and benefits are very evident, a large number of its drawbacks are underestimated, and these may be critical when deciding to apply blockchain to a supply chain. [59]

2.4.2.1. Technical limitations and concerns regarding scalability

The technical constraints incorporate, yet are not restricted to throughput, latency, size and bandwidth and protection.

- **Throughput**

  Current blockchain technologies, have a high throughput, however not as high as certain centralized system, even in private deployments for example, Corda. This is one of the primary worries in the supply chain, that the data can’t be processed by a blockchain as quick as the current systems, which could prompt to low performance and further delays. Nevertheless, as in any distributed system, the reduction in speed is always the payable cost for decentralization in the supply chain. At last, while the progression of data within a particular organization may be lower than previously, the flow between various organizations, which had not been previously coordinated, may really be a lot quicker than previous. Comparing the speed of a centralized system with the speed of a distributed ledger isn’t really fair, since the last provides greater functionality and further disperses the data where the previous couldn’t [59]. In this case, slower dissemination of data globally, across different organizations, is preferable to a quick dispersion just locally, through network of an entity or its nearest associates. In the end, a trade-off among efficiency and functionality may occur, with higher functionality and integration being accomplished at the expense of output.

- **Latency**

  Also to throughput, and even linked with it too, a transaction latency is something to consider. With some blockchain deployments, like Bitcoin, may take a long time for each transaction to approve, and this time may also rely upon the fees paid [59]. This is mitigated by permissioned systems, for example, Corda, which have low latency, even within the sight of a high number of transaction, and have no currency or charges to worry about.

- **Size**

  The more transactions are processed and the more data that is held in the blockchain, the greater it really increases. In the current sense, if we somehow happened to send a worldwide blockchain for all the supply chains, it would likely develop far too enormous in a short period of time, which in the long run would not be supportable. However it would not likely be as large of a problem in an increasingly constrained scope [59]. There is also a lot of research on blockchain size optimization.

- **Security**

  One worry for blockchains, is the way security is taken care of. In the case of public blockchains with PoW consensus this problem is more significant There are several other options in the case of supply chains, and maybe using a public PoW blockchain isn't the ideal one, so that’s not the fundamental security thing to be concerned about. Nonetheless, there is another thing to consider, which is the possibility that the hash function being used right now may be broken in certain years [59]. Unless somehow happened, a blockchain’s immutability property would be violated and any possibilities of demonstrating the provenance or traceability of products would lose their groundwork.

2.4.2.2. Lack of interoperability standards

Provided that there is an approach to exchange data, each organizations has their own methods for integrating data into their systems, in whatever type they need, as long as there is appropriate pieces of data. Many organizations which should get to this data can have troublesome in knowing exactly what to search for and where to search for it. [59]

That is the initial part of the issue, the absence of principles that organizations can abide by if they want to establish a shared ground by which the can participate and see each other's data.

The second part of the issue deals, in a progressively specialized way, with the absence of interoperability in the system’s themselves. Numerous Enterprise Resource Planning (***ERP***) systems work in a shut environment, with data regularly being entered manually in the system, and no outside systems APIs to link to.

2.5. Similar existing applications

Several initiatives that attempt to suit the blockchain as a solution to boost SCM have just risen, or are in the works. Several of these applications are discussed in this section.

2.5.1. CargoX

CargoX offers an answer for digitalization of Bill of Lading (B/L) papers. To give some unique circumstance: the items are delivered via cargo ships, inside containers, many times in a supply chain. The B/L record has a similar value as the items are declared on it and serves the functions below [53]:

- It is a receipt which recognizes product loading.
- It contains the conditions of a carriage contract.
- It is the title to the products which it declares. Such feature make it an

incredibly important document that must be moved from the transporter to the organization that safely acquires the items. Losing this record would mean losing the rights to all the products in the shipment, just as losing the evidence that they were even shipped in any case.

CargoX uses smart contracts from Ethereum to put those paper documents in the blockchain. It has an integrated token system which permits the exchange of the possession of documents immediately after payment [53].

2.5.2. Eximchain

Eximchain is an inside and out solution, that function as a ledger, recording historical information, and transactions, as a stock management tool and furthermore offers smart contracts for financial applications. It was created utilizing an Ethereum fork, Quorum, which is a permissioned version of Ethereum, based on enterprise use, meaning Eximchain operating on its own network [54].

Eximchain smart contracts consider the transmission of funds and confirmation of the legitimacy of placed orders. All information from these transactions, including the shipment of products, are registered on the ledger, permitting providers to demonstrate their distribution reliability.

Simultaneously, the stock, just as all the products, are tracked with a small delay across the supply, with the data being smoothly accessible among partners. This permits for greater accurate prediction of supply and request expectations, particularly if the data is combined directly with predictions systems.

2.6. Designing a blockchain-based supply chain

Blockchain isn't necessarily a one-size-fits-all solution, and its utilization must be specifically custom fitted to the particular requirements and application in question. The area discusses several significant reference points to keep in mind to take decisions about building a supply chain based on blockchain.

2.6.1. Integration models

- **Point-to-point**

  Business-to-Business (B2B) Data Interchange - The mix between every two specific endpoints must be designed. Each new association has to be individually modelled. It doesn't work well in an enormous scale, it is a model that works only under particular cases and necessities because it requires custom integration.[42]

- **One-to-many entities**

  Hub point B2B - An organization can build up a network endpoint that can be linked to different organizations as long as they keep the communication requirements of the hub or utilize its API. That way, a single organization can communicate with different mediators.[42]

- **Many-to-many entities**

  Cloud B2B - This model provides complete integration where the data can flow freely between organizations. It is a public blockchain definitive objective, however it would allow organizations the establish interoperability standards that are currently missing. Else, it is the most cost-effective model and the one that can achieve the most benefits, provided that the organizations can improve their services to integrate with the blockchain.[42]

2.6.2. Key implementation components and features

As shown in the recently referenced projects, such as CargoX, Eximchain, every one of them pursued their own aim and objectives, and every one of them has a specific set of features that permits them to accomplish these objectives. Likewise, below are some of the key parts that a blockchain has, and the respective characteristics that must be agreed on:

- **Information storage**

  A blockchain most fundamental and critical feature is its ability to save information, which is then viewed as changeless, just as enrolling any significant events. As such, storage of data may be essential in the supply chain [42]. It additionally allows Inventory Management conceivable (however its usage is beyond the scope of the blockchain), traceability and product provenance.

- **Ledger and transactions**

  Similarly, it may be necessary to enable transactions and to record them into the chain, specifically in what accounts for payments between companies.[42]

- **Smart contracts**

  At last, smart contracts can possibly be a significant part of SCM. Smart contracts were utilized in the applications we mentioned to move ownership of information or items through tokens. That is only one of the various uses that are potential [42]. Certain smart contract utilizes incorporate tracking objects by their position or state, automatically refreshing the status on the blockchain, and responding by telling the company responsible for the objects to any significant event. Automation of payments upon delivery is another potential capability.

  Finally, smart contracts permit for practically any application, because they are code, programs being run on the blockchain, thus, they are one of the elements of blockchain with the most potential for new and creative features to be created upon.

3. Internet of things

3.1. Definition

Internet of things is one of the emerging technologies allows people and objects to be interconnected through the internet, recent statistics estimate [55] billions of objects are already connected to the internet. Objects (things) takes the form of small things such ***Nano*** Internet of things to ***smart cities*** [57] and ***industry*** [58], and many applications in different domains such healthcare, education, and energy trading, like the application scenario discussed in [56].

3.2. Architecture of the IoT model

From a technical and architectural point of view, IoT can be built based on several protocols organized in three main layers ***Figure 2.3*** : the data perception layer, the network layer and thirdly the application layer. The figure below illustrates such an organization.

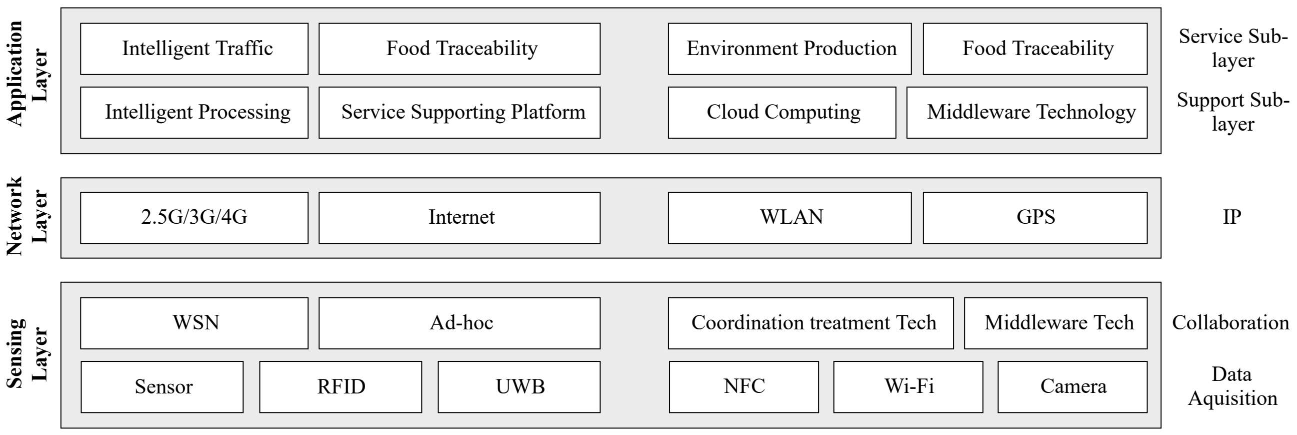


Figure 2.3 : The Iot architecture model in the vertical view.[61]

3.2.1. Perception layer (sensing layer)

Two main features, data acquisition and collaboration, are considered here. The event or state of an "Object" in the physical world such as temperature, concentration and multimedia data has been perceived and acquired by detection devices, such as sensors, ***RFID*** (Radio Frequency IDentification) Beacons, cameras and GPS terminals. Advanced techniques in this layer focus on designs and implementations of new sensors with low power consumption and high performance, the technology integrates all short-range communication technologies, such as ***RFID, UWB*** (Ultra- broadband), ***NFC*** (Near Field Communication).

3.2.2. Network layer

This layer is responsible for the reliable transmission of data generated in the perception layer as well as ensuring connectivity between connected objects and between intelligent objects and other Internet hosts. On the other hand, it is expected that the data from the physical layer (Device) will be enormous because the number of objects connected to the Internet continues to increase rapidly.

Number of issues still to be addressed to meet the new demands of IoT Applications:

- **Addressing**

  Each "Object" in the IoT is only associated with an address in the digital world. Due to the large scale of IPv4 in the Internet, converting from IPv4 to IPv6 would be a lengthy process, which raises the issue of compatibility.

- **Network integration**

  IoT is a large-scale, complex, real-time and heterogeneous network. Several heterogeneous terminals are heavily deployed in space. The strategies for Network integration is fusion and collaboration.

3.2.3. Application layer

This layer includes the support sublayer and the service sublayer. the Support Sublayer delivers results based on user demands, the system in this sublayer, integrates distributed computing technologies, such as P2P (Peer-to-Peer) and cloud computing, both of which facilitate Analysis and processing, decision making and strengthening of the information capacity of processing in the IoT. Based on the Support sublayer, the Services sublayer provide interfaces and platforms from an extensible service framework.

3.3. Networking in IoT

Networking is one of the important part to be considered in IoT, there are several technologies provided to connect devices to the internet, in this section, we focus on the following standards used on IoT communications, ***WiFi*** (or cabled), ***Cellular***, ***LPWAN***, and ***LoRa***[60].

3.3.1. Wireless or cabled network

Considering that the IoT is an extension of the internet, Wi-Fi is deemed the most common communication standard in the internet, devices are able to connect to an IEEE 802.x network, several modules are available in the market, such ***Expressif-8266***, ESP-32, and Galellio ***figure 2.4***. these technologies are relatively cheap and could be considered as a good option to extend or benefits from an available network, moreover, it supports a high-payload data transfers.

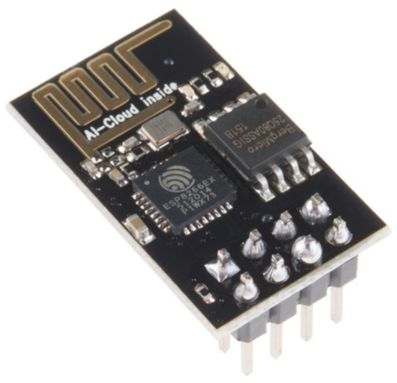
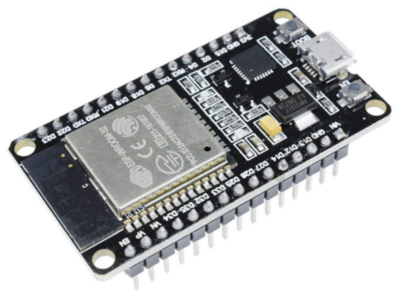
Expressif-8266	**ESP-32**	**Galellio**
		

Figure 2.4 : Some models devices available in the market.

3.3.2. Cellular/mobile network

Currently deployed cellular networks such as GSM, CDMA and LTE, the basic idea is to split the territory into a number of cells or zones, in each cell the connection is maintained by a base station (i.e. ***BTS*** or ***eNodeB*** or ***eNB***). and transceivers, mobile subscribers are connected through RF frequency to provide many services such as voice/data services. This solution could be an alternative solution when the device does not support the Wi-Fi communication or even if the device is not in the range of the Wi-Fi network. another advantage is that the mobile and fixed subscribers can connected immediately with cellular network as soon as mobile phones are switched on. Since all the handshake signals between mobile and base station are automatically exchanged. However, this type of networks offers less data rate compare to other wired and wireless networks, Moreover, As the communication is over the air, it has security vulnerabilities, climatic conditions. In term of cost.

3.3.3. Low-Power wide area network (LPWAN) or low power WAN

Another type of wireless network designed to work in wide range area (long-range communication) at very low-power, low bit rate (0.3 Kbit/s to 50 Kbit/s per channel), and low- frequency, which make it useful to connect small objects where the data exchange is really very small. The number of message or data payload that could be transmitted by LPWAN networks are limited, many LPAWNs providers are available in the market, we cite the example of ***SigFox***. LPWAN do not have a direct connection from the device to the internet, instead, it relies on existing networks using web services to connect to the designed applications. Moreover, instead of providing internet connection to the IoT device, LPWAN provides tools and functions to create triggers received from the devices to the network.

- **SigFox**

  Sigfox rolls the first worldwide 0G network to listen to billions of objects transmitting data, without the need for network links to be built and maintained. This unusual approach in the field of wireless communication, where there is no overhead signaling, a minimized and upgraded protocol, and where objects exchange not connected to the network.[62]

3.3.4. LoRa or LoRan (Long-range network)

LoRa is considered a sub class of LPWAN network, the difference is that instead of using a service provider, devices can use a gateway to access the internet, ***ZigBee*** is one example of LoRan networks.

- **ZigBee**

  ZigBee is an alluring technology for deploying low-cost, low-power wireless control network requiring high adaptability in node location. The technology is defined by ZigBee specification kept up and published by ZigBee Alliance. ZigBee is a specification based on IEEE 802.15.4 that specifies the ZigBee PHY layer and MAC layer for making personal area network (***PAN***) with small, low-power digital radios in 900 MHz and 2.4 GHz. ZigBee also specifies the architecture for the network, security and application framework for an IEEE 802.15.4-based system. [63]

3.3.5. IoT networking recapitulation

Network	**Wi-Fi**	**Mobile**	**LPWAN**	**LoRan**
Speed	High (1 Gb/s)	High (41Mb/s)	Low (50Kb/s)	Low
Payload	High	High	Low	Low
Range	Low (150m)	High (30km)	High (10km)	High (45km)
Connection initialization	Bidirectional	Bidirectional	Device	Bidirectional
Cost	Low	High	Low	Low
Infrastructure	Private/public	Provider	Provider	Private
Example	Smart Home	Smart Watch	Smart Irrigation	Cattle Tracker

Table 2.2 : IoT networking recapitulation.

Conclusion

During this chapter , we mentioned the most important details about the mechanism of the Internet of Things, the supply chain and smart contracts, as well as examples of applications for each of them, and as a result of this chapter, we came to list the points and common ties between the three technologies that allow us to merge and open new horizons while maintaining all of its advantages, which we will touch upon the last chapter.

Chapter III

Milk supply chain, Implementation and Comparison.

Introduction

In this chapter we will introduce dairy production supply chain recognizes some of the basic concepts and use it as our study for comparison between private and public platforms, we mentioned how does dairy production supply chain and scenarios and problems in it and the use of IoT in this supply chain, and motivation for choosing blockchain platforms Ethereum and Corda and their definition and architecture and a comparison between them without going into the specifics of the dairy industry in depth.

1. Milk supply chain

1.1. Overview

Milk has been used by humans since the beginning of recorded time to provide both fresh and storable nutritious foods. Almost half of the milk produced is consumed in some countries as fresh pasteurized whole, low-fat or skim milk. Most milk, however, is processed into more steady global dairy products, such as butter, cheese, dried milk, ice cream, and condensed milk [64].

Due to rising population , increasing incomes, urbanization and the westernization of diets in countries such as China and India, global demand for milk is predicted to expand (by around 60% in 2050) to a large extent. Around the world, there are about 270 million dairy cows. 811 million metric tons of milk were processed in 2017, and annual consumption of milk products (fresh milk equivalent basis, except butter) per capital around the world is predicted to increase by more than one-third on a per capital basis by 2067 [65].

1.2. Milk supply chain process (workflow)

Milk supply chain refers to the movement of milk and dairy goods from suppliers to the table of customers, such as raw materials → production → sales. By easily transferring milk from a dairy farmer to the feasting table, more reliable supply can be obtained.

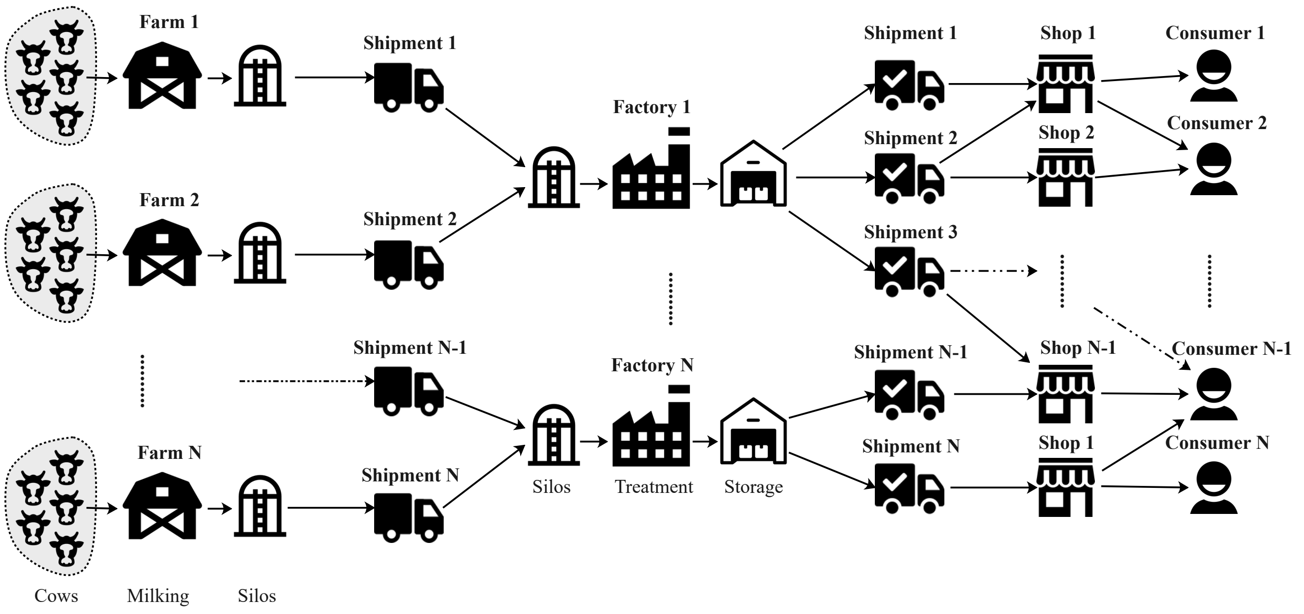


Figure 3.1 : Milk supply chain.

The dairy supply chain has become a complex network of steps involving each of the farms, suppliers, platforms of sales, etc. There are many opportunities within this complicated and advanced network to include strategies for protecting and covering threats occurring across this network. The dairy supply chain starts from farms with cow's milk till it reaches as a product that can be sold to the customer, and is one of the most critical processes for preserving human survival worldwide. The food chain system represents 10% of global gross national product, according to the World Bank Report.

But the process of producing and distributing milk that starts from cows on farms, followed by processing operations in the factory and then distributing it to the stores before the end of the trip at the user's table, it is expected that, with this complexity in the supply chain and the important processes included in this system, the dairy supply chain will become a venerable of several Various and vital risks.

We will suggest this scenario with the following different parties and their respective roles in the previous supply chain :

- ## Farm

The best types of cows producing milk are raised and the safety and health of the cow is maintained, as medical tests are carried out regularly before each milking process and after the udder cleaning process to ensure that it is healthy and free of any infectious diseases.

Cow milking is performed in batches twice a day for at least 10-15 minutes and the milk is preserved for up to two days after filtering at temperatures not exceeding 4 degrees Celsius from any impurities in refrigerated silos in order to avoid the reproduction of any form of microorganism.

The milk is transported to the processing factories by transport trucks supplied with a cooling system after checking the percentage of water in the milk and the temperature not exceeding 4-5°C.

- **Factory**

  After the arrival of the milk through the transport trucks, the percentage of water in the milk is first checked, which is about 87%, with some laboratory tests for the milk that do not take 15 minutes. If the tests are not passed successfully, the shipment will be rejected.

  In the case that the milk shipment passes the tests, it is transported from the trucks to the factory silos, which in turn store the milk at a temperature less than 4°C in preparation for the pasteurization process.

  The milk is pasteurized inside the sterilized factory in a very strict environment, and the milk is packed into pre-prepared boxes with the manufacturing and end-of-life information added to it, then the cans are collected in small units and transported by the panels to the factory's giant refrigerated warehouses.

  After the loading plates have remained in the warehouse for less then 7 days, they are transported and distributed to the stores according to the information presented in a tape on each plate in small refrigerated trucks.

- **Shop**

  Milk is received in units directly from the factory by small refrigerated trucks and is added to the refrigerated shelves in order to be sold to the final customer in the next few hours as a final step for the product.

- **Consumer**

  the consumer is the final station for the processed milk after it has been transferred from the farm to the factory and then followed by the store, and the information in the boxes helps the customer to track the stages of manufacturing and the transfer of the product at the end of its arrival to the customer.

2. Problem statement

It is expected that the milk production and distribution process, which were represented in the previous supply chain ***Figure 3.1***, is exposed to many risks and problems due to its complexity and sensitivity of the product (milk).

As the poor health of cows due to negligence of their regular medical examination and the failure to provide the appropriate environment in time for milk may be the guarantor of the failure of the product, and in turn break the entire chain. Not to mention that it is not possible to know at what part of the chain the problem has occurred due to the parties avoid responsibility for logistical errors, from which the small and weak parties bear all losses, which will create a kind of distrust between the parties In turn, it will disrupt and break the entire chain.

Another problem is the lack of transparency in transactions, and thus the lack of a transparent pricing system in light of the industrial giants companies trying to exploit farmers to achieve greater profits, due to farmers ’ignorance and insufficient information about market requirements for milk that leads to allocating Ineffective materials, including a surplus in production, as happened in the United States in 2020 during the period of the global Corona pandemic, which led farmers in the state of Wisconsin to pouring 25 thousand gallons of milk in the sewers[66], The question here may be ***"How smart contract be efficient in dairy production?"***

3. Related work

In this section, some ideas that been suggested or working with will be addressed, to manage dairy supply chain using available technologies, and mentioning some real life examples and pros and cons of each one of them.

Traditional supply chain : It considered as one of the oldest supply chain based on human interference who manage the chain and save data manually, the traditional supply chain is still used in a lot of countries,it considered old and unreliable, its main short comes :

- Fully reliance on the human component, which lead to increase of intentional and unintentional faults, also transparency is not guaranteed due to deceiving people who would take advantage of farm or factory ignorance to make profit on transaction made.
- Monitoring of cows is done manually, which will cause a kind of delay in responding to diseases, especially in large herds.
- Ensure that the environment for transporting and storing milk meets the necessary requirements(temperature) manually, thus increasing the potential for contamination of milk.

Supply chain IoT-based : The introduction of the Internet of Things into the supply chain is not anything new, particularly in the field of perishable food, but the advancement of the devices used, such as sensors and systems themselves, has opened the way for us to introduce a range of ideas in the recent past that have only been superficial ideas.

We mention some of the uses of IOT bellow :

- ***Monitoring the health of cows and milk***

By installing smart chip within cows with identification numbers, sensing vital indicators such as cow temperature, etc., and sending periodic reports either to the farmer or to the outside parties, these chips can allow to isolate and help sick cows as soon as possible to avoid any problems during the milking process and not mixing any healthy cows milk with any sick cows milk, this will serve as a guide for other parties in the supply chain to ensure the validity and quality of milk.

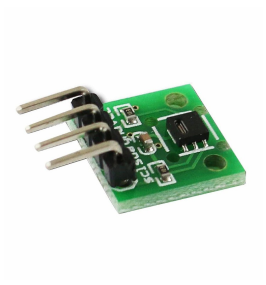
Figure 3.2 : DairyMilk M6900 - Cow Sensors.[82]

- ***Automatic analysis at every production stage***

To do the study at any point of production: This begins with the food that the cows consume, whether it's the grass in the field or whatever. Both aspects of the animal continue to be tracked by this study and then continue at the processing plant. Offer milk monitoring systems through sensors that measure the physical and chemical properties of milk, using optical sensors, conductivity, flow and thermal sensors with milking machines.

Figure 3.3 : Milk Analyzer.[83]

- ***Tracking milk and changes happed to it***



The two stages of milk transport and storage are considered to be one of the most sensitive stages, as milk becomes more sensitive to change, so installing sensors connected to the system in transport trucks and silos is sufficient to prevent any quantity of milk from reaching the second party that did not meet the required conditions and all this automatically and without human interference.

Figure 3.4 : SHT20 temperature and humidity sensor.[84]

Choosing the architecture on which the system will be developed in, is considered to be one of the most important and sensitive steps in any enormous project in which millions of dollars are exchanged on a daily basis, as it needs a high degree of security and reliability, And the supply chain, which is known for its huge operation, we need a system capable of dealing with all events to be provided effectively, there are mainly two models of handling supply chain issues:

Centralization-based solution : One of the most common and easiest models for creating an interactive computerized system, since it represents a central server that is distributed to the service (Government Agency) and is considered a hub for all transactions in the system. As for clients, they are The Interactor's or service-seekers (Farms, Transport, Factories, etc.). When talking about a server-client based system model, it means a central system,we have this Dairy Supply chain based on client server model [67] ,Which in turn, contains some many weaknesses that will cause our supply chain to collapse, and the most important of these weaknesses:

- If all the clients at the same time request information from the server, it's going to get overloaded. This could cause congestion in the network.
- If the server fails for any reason, then none of the requests of the clients can be served. This causes the client server network to fail.
- The cost of setting and maintaining a client server model are very high.
- Furthermore, due to the centralization of the system, a very critical aspect, namely reliability, would be missing.
- It exist third authority that validate the transaction between two parties (two clients), which is the server and the only one responsible for all operations on the system and has full control of it.

P2P model : Which is a collection of devices that store and share data simultaneously, and P2P architecture is ideal for different applications , particularly for systems that rely on the continuity of their network, and there is some Big Tech companies who used supply chain in P2P Network such as Walmart [68] Food Supply Chain and also K&G [69] Logistics advancement using P2P Network and more. OriginTrail [70] created Wine Tracing with smart sensors ,and one of the most important advantages of the P2P Model,which was a key factor in our decision as the architecture on which our project will be developed, is :

- Each node in the peer to peer network manages itself. Therefor, the network is kind simple to set up and maintain.
- In the client server network, the server handles all the requests of the clients. This provision isn't needed in peer to peer computing and also the cost of the server is saved.
- It is simple to scale the peer to peer network and add a additional nodes. This only increases the data sharing capacity of the system.
- None of the nodes in a peer-to-peer network depend on the other nodes to operate them.
- And, because of the decentralization of the system, it will provide us with a very important element of reliability and transparency.

4. Realization

To implement and build a system capable of managing the dairy supply chain We chose P2P as a model, due to all the advantages it offers, in particular security, and our choice of blockchain technology, which is the main objective of this study and which we addressed in the first chapter on its types and characteristics, will now ask the following question.
"What kind of blockchain is suitable for our application?"

4.1 Blockchain platforms

This study will focus on the distinction between two platforms, Ethereum and Corda. Ethereum was chosen because it currently has one of the biggest follow-ups from users and developers on the blockchain scene. There are currently more than 2,500 applications developed on top of the Ethereum[71], the highest among similar platforms. The Ethereum Development Team said the aim was to provide general tools for a variety of applications and to make it simple enough to gain popularity.

Corda was chosen because it is a distributed ledger technology supported by a number of diverse financial institutions and banks such as JPMorgan, Bank of America and also Nordic banks such as SEB and Nordea [72]. This makes it important since most of the previous blockchain networks promoted themselves as decentralized platforms to cut banks and other middlemen out of the transaction equation.

4.2. Ethereum

Ethereum is a blockchain platform which was first made public in January 23, 2014 when the founder Vitalik Buterin announced it in their blog [73] .The goal of the platform is to offer alternative protocol for developing decentralised applications. This is accomplished by abstract foundational layer built by Ethereum’s team which is based on blockchain with built-in Turing-complete programming language Solidity. This allows anyone to easily develop decentralised applications and smart contracts where they can define rules for ownership, transaction formats and state transition functions [74]. All the code is compiled down to Ethereum Virtual Machine and deployed to the blockchain for execution.

4.2.1. Architecture

The Ethereum architecture follows a few of the fundamental design concepts: simplicity, universality and modularity [74]. This means that the Ethereum team has made the platform as simple as possible in the expectation that developers will achieve wide acceptance. The platform is said to not concentrate on built-in functionality, but to include the internal scripting language tools to develop any kind of dapp.

The "Account" is one of the main concepts used in Ethereum to describe the state. And account has an address, and all state transfers simply mean that a value and information is transferred between accounts. An account is composed of four fields:

1. The nonce.
2. Current ether balance.
3. Contract code (only if it is contract account).
4. Storage (empty by default)

4.2.2. Nonce

Any transaction has a nonce in Ethereum. Nonce refers to the number of transactions sent from the address given. The nonce value increases by 1 for each transaction, which also avoids double-spending as the nonce would always define the order of transactions [75]. Usually, if a double-spend occurs, it is because of the following process:

- A transaction is sent to one party.
- Another transaction is sent faster with a high gas price.
- The second transaction is mined first, therefore invalidating the first transaction.

There are rules on what transactions are considered valid transactions, and some of these rules are implemented using nonce. Particularly:

- **Transactions must be in order :** You cannot have a transaction with a nonce of ***1*** mined before transaction with a nonce of ***0***.
- **No skipping :** You cannot have a transaction with a nonce of ***2*** mined if you have not already sent transactions with a nonce of ***1*** and ***0***.

4.2.3. Gas

The blockchain for Ethereum is a network. The fuel for that network is ether (ETH). You have to pay for the computation when you transfer tokens, interact with contracts, or something else on the blockchain, regardless of whether the transaction is successful [75]. or not The Gas Limit Gas Price is the overall cost of a transaction (the "transaction fee").

- **Gas Limit**

  The gas limit is referred to as the limit because it is the highest amount of gas units you are prepared to spend on a transaction. This eliminates cases where somewhere in a contract there is a mistake [75].
- **Gas Price**

  If you want to spend less on a transaction, you can do that by decreasing the price you pay per unit of gas. The price you pay for each unit increases or decreases the rapidity of the mining of your transaction [75].

4.2.4. Contracts

Solidity is the most common and most widely supported option in the languages of smart contract [74]. Smart contracts in solidity are much like object-oriented programming classes. There may be state variables, functions, events, etc. in each contract.

4.2.5. Consensus

Ethereum is currently using **PoW** consensus algorithm that relies on computing power to mine blocks, but the first version of Casper (Ethereum 2.0), a nickname for a much awaited update, has been published and Ethereum is a step closer to switch to PoS algorithm [76]. And PoS relies on the ***stake*** (typically the amount of currency a user holds) of the node in the system. The more a user owns a stake, the more power they have over validation [77].

4.3. Corda

Corda is a distributed ledger platform with the main purpose of managing and documenting financial agreements. The Corda consortium was created on 15 September 2015, featuring nine of the world's largest banks, featuring JP Morgan, Goldman Sachs and Barclays [78]. The project itself was made publicly accessible and opened on 30 November 2016 [79]. The goal of the project was primarily to reinvent financial data and agreement management structures. The Bitcoin and Ethereum platforms that came before Corda, while not really specifically applicable to financial institutions, enabled the team to consider a new way of creating distributed systems. Corda Platform also supports smart contracts that are automated or can work with human input and control. The main idea here is that cordas Smart Contracts will represent rights and responsibilities expressed in legal terms and could be legally enforceable [80].

4.3.1. Architecture

Corda network consists of five major components:

1. Nodes that communicate with TLS through the Advanced Message Queuing Protocol (***AMQP***). Nodes provide a relational database for the storing of data.
2. Permissioning service for ***TLS*** (Transport Layer Security) certificates.
3. Network mapping service, that is used to publish on the network node information.
4. One or more notary services, which may be spread over a number of nodes.
5. Zero or more of the oracle services. Which allows the ledger to link to the real world by signing transactions if the facts mentioned in them are considered to be valid.

4.3.2. The network service

The way R3 Corda operates is very different from the conventional Blockchain networks. In public Blockchain networks such as Bitcoin or Ethereum, anybody may join the Blockchain network and will only be known by a pseudonymous public address. Messages are also sent to the network in a raw form. This architecture may not be appropriate for companies where security and privacy of messages are essential. In the Corda network, each node is a verified IP address that is agreed between business counterparties previous to the conclusion of the contract. These IP addresses representing various companies go through a strict KYC process and are linked to the network through a network map service called the "doorman." All nodes may use this network map service to communicate with other nodes in a peer-to-peer way. [81]

Unlike public Blockchain networks such as Bitcoin and Ethereum, where broadcasting or gossip networks are in operation, in Corda, nodes connect on a peer-to-peer basis with Transport Layer Security (TLS)-encrypted messages sent over AMQP/1.0.

4.3.3. Identity

As noted with the second part, corda is a permissioned blockchain network. The identity of each group is known and used when interacting with others. Connection to the network is often managed by a doorman.
Corda is connected to the identity of the nodes that actually stand for the individual organizations that operate in the DLT network. Therefore, each node arrives with its own legal name, IP address and an X.509 certificate signed by the doorman. Identity in the Corda network may reflect an organization or service that assists the DLT network. [81]

4.3.4. State

The data in Corda is stored by state objects. The data state is immutable; but, the state could change with each transaction and we still can verify the previous state of the data for traceability. Each node holds a local database called "vault" which holds present data (also known as unconsumed state) as well as previous data (called consumed state) with timestamps. Each vault has several different representations of data in the form of state objects, of which only one is present and the others are historical data. [81]

4.3.5. Transaction

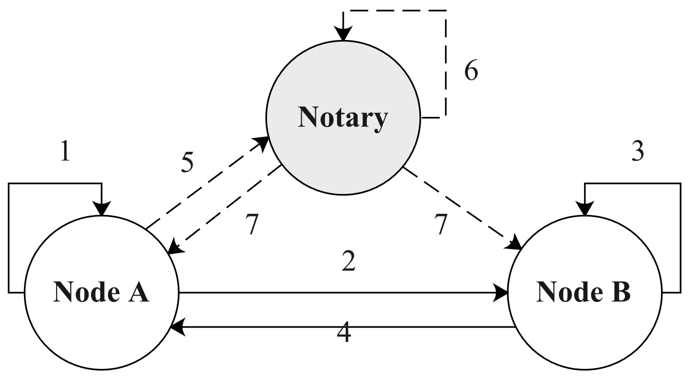
Corda describes a transaction as a "proposal for updating the ledger" that will be made if it is contractually valid, that is to say, signed by both parties concerned and notary [81]. Last but not least, there should be no double spending. Corda uses a ***UTXO*** (unspent transaction output) model where a transaction may have between zero and several inputs and outputs.

Transaction inputs can be any of the following:

- Command
- Attachment
- Timestamp

4.3.6. Flow

Flow is a series of steps that informs the node how to make a particular update to the ledger. Flow provides contract validations as well as state changes [81]. And the communication mechanism in Corda can be represented as follows



1) Generating, verifying, signing transaction
2) Gathering counter party signatures
3) Verifying & signing transaction
4) Sending signed transaction to initiator
5) Requesting notary signature
6) Verifying transaction by notary
7) Broadcasting transaction to participants

Figure 3.5 : The communication mechanism in Corda.

4.3.7. Contracts

Smart contracts in Corda can be written in ***JVM*** languages as Kotlin or Java. and for every transaction to be executed must be signed by each of the notaries involved and must also be contractually valid [81]. There is a one-to-one agreement between the state and the contract.

4.3.8. Consensus

In Corda, only associated parties and also notaries engage in the transaction, and there is also no necessity for consensus from other nodes, like most other Blockchain platforms. The involved parties verify the validity of the coded contract and the uniqueness of the transaction [81]. And the communication mechanism in Corda can be represented as follows.

4.3.9. Notary

A notary is a special node that participates in transactions, validates them and avoids double spending. The Corda network can have as many notaries with different roles as follows: [81]

- **Privacy :** Corda may have several validating and nonvalidating notary services on the same network operating with different algorithms. The node will programmatically select a notary on a per-transaction basis.
- **Load balancing :** We can also spread transaction loads over several notaries, allowing a higher transaction throughput to the platform overall.
- **Low latency :** A notary who is physically closer to the parties to the transaction may be selected over others to reduce latency.

4.3.10. Oracles

Applications deployed on DLT as Corda are deployed in a private permissioned network that is fully cut off from an outer network. But consider the possibility that there is a need to link to some external service to retrieve data such as location information, exchange rate, and so on? For that, we use the services of Oracle. [81]

4.4. Tools and developing environments

4.4.1. Software development environment

- **IntelliJ IDEA**

IntelliJ IDEA is an integrated development environment (IDE) written in Java for developing computer software. It is developed by JetBrains (formerly known as IntelliJ), and is available as an *Apache 2 Licensed* community edition, and in a proprietary commercial edition. Both can be used for commercial development.
Figure 3.6 : IntelliJ IDEA.

- **Visual Studio Code**

Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.
Figure 3.7 : Visual Studio Code.

- **Geth**

Go Ethereum is one of the original implementations of the Ethereum protocol. It is written in Go, fully open source and licensed under the GNU LGPL v3. **Figure 3.8 :** Geth.

- **Remix IDE**

Remix is a browser-based compiler and IDE that enables users to build Ethereum contracts with Solidity language and to debug transactions. **Figure 3.9 :** Remix IDE.

4.4.2. Developing languages

- **Solidity**

Solidity is an object-oriented programming language for writing smart contracts. It is used for implementing smart contracts on various blockchain platforms, most notably, Ethereum. **Figure 3.10 :** Solidity.

- **Kotlin**

Is a functional, object-oriented programming language with static typing. It runs under the Java Virtual Machine (JVM) and can also be compiled under JavaScript, in the same way that the Java language takes its name from the Indonesian island Java, the name Kotlin comes from the Kotline island off the coast of St Petersburg in Russia.
Figure 3.11 : Kotlin.

4.4.3. Hardware configuration

During this present project, all the work has been done on two computers (laptop and desktop) which have the following technical characteristics.

Materials	**Laptop**	**Desktop**
CPU	Intel Core i5-7200u @ 2.50 GHz	AMD Ryzen 5 1600 @ 3.20 GHz
RAM	8 GO	16 GO
GPU	AMD Radeon R7 M440	NVIDIA GeForce GTX 1050ti
OS	Windows 10 Pro x64	Windows 10 Pro x64
STORAGE	HDD 1Tb 7200RPM	XPG S40G M.2 NVMe SSD

Table 3.1 : Hardware Environment.

4.5. System design

We will present ***Figure 3.1*** in the next diagram class (*Part used in this study*) to make it simpler how we coded this chain considering the minor variations between the two platforms, but the concept remains the same.

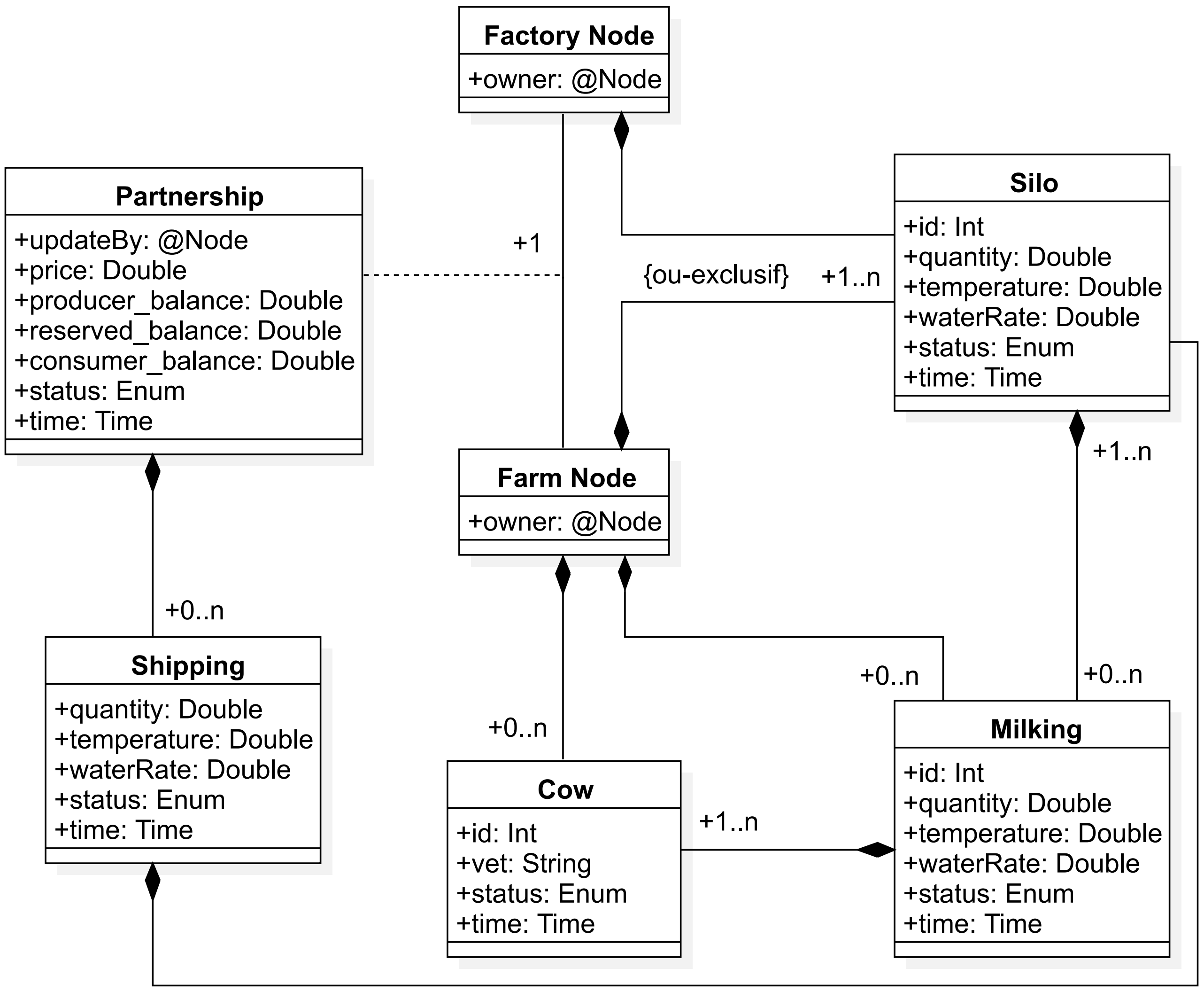


Figure 3.12 : Class diagram of supply chain.

As shown in ***Figure 3.12***, the network contains many parties with various roles and the functions to be taken in consideration. For example, only the node "***farm***" has the authority to create and change cows. In addition, a partnership contract must be created before any trading activity between two nodes. Where an agreement is established on the pricing (liter price) and determining the producer and the customer in that agreement, apart from the customer putting a significant amount of money into that partnership, to facilitate logistic operations between the two parties without the intervention of a third party (***Bank***) in the transaction, while at the same time maintaining the right of each party to make the transaction automatic, especially after the partnership include two parties just to not interference in transactions. It is mentioned that each node is capable of creating ***2(N-1)*** partnership contracts in the network with a different role from customer to producer dependent on transactions (not discussed in this research).

4.6. Contracts implementation

Simplifying the milk supply chain represented in ***Figure 3.1*** is an important step towards creating a simple model before starting work on the programming process, and it is also important to identify the contracts within the supply chain and the role of each of them, and this in itself is a challenge for the different geometries of Ethereum and Corda. ***Figure 3.13*** represent a simplification of ***Figure 3.1*** as follows:

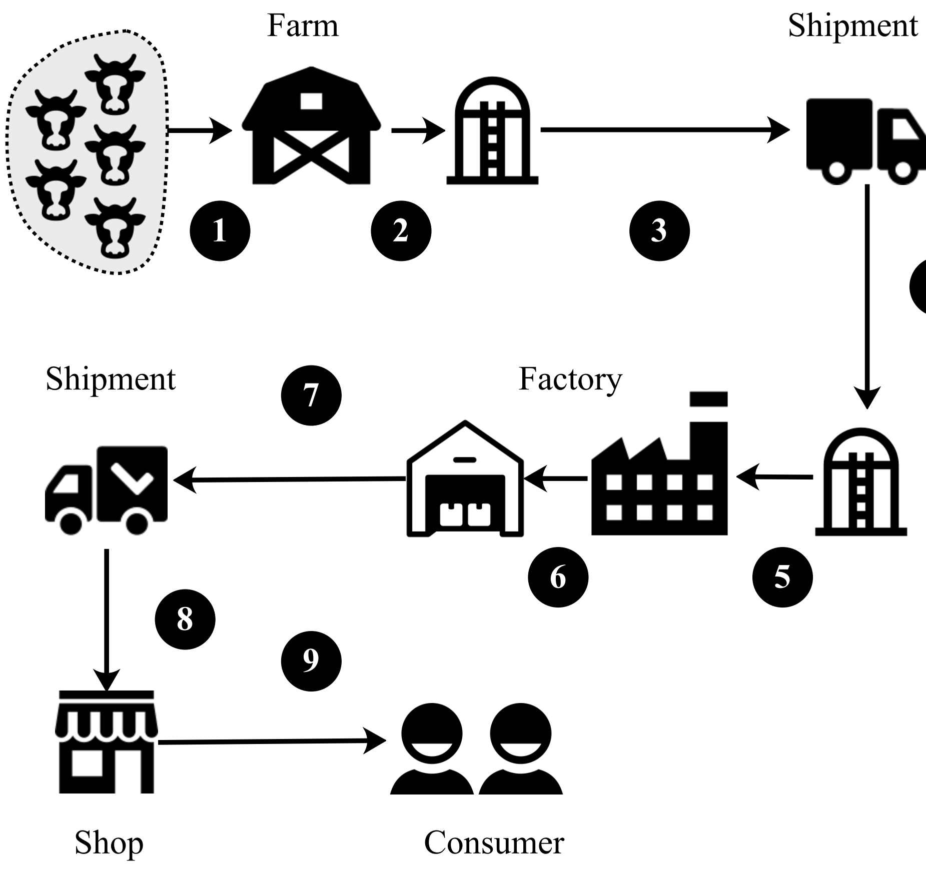


Figure 3.13 : Simplified milk supply chain and Smart contracts.

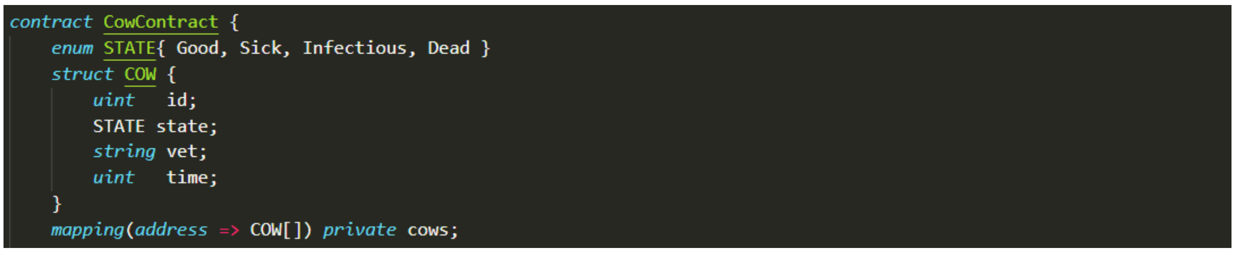
Although the contracts shown in the picture do not reflect the main contracts in the chain, we will not explain each one of them in this paragraph, but will explain, step by step, two examples of the differences and similarities between Corda and Ethereum.

4.6.1. Deploying contracts

- **Milking cows (see contract No. 1 in Figure 3.13)**

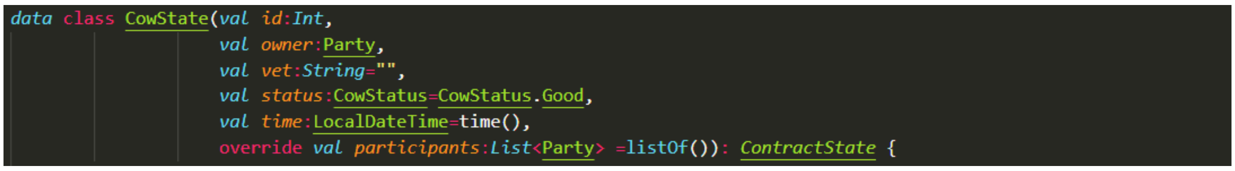
Cows are known in the blockchain network by their identifier and owner or farm ***Figure 3.12***, and they are treated as a class (contract) by themselves. As the sensor updates its information regularly and in real time within the network, the sensor can send several types of data, such as its geographical position and cow health etc. , which can be of great benefit to the farmer. for us and the supply chain, Cow health is the most important thing on which the supply chain is founded, and because of the various types of diseases that cows are exposed to from infectious and non-infectious diseases, we represent their health status by enum class named ***CowStatus*** with the following values: ***"Good, Sick, Infectious, Dead (used for history)"***, and added the possibility of updating the health status of cows by a veterinarian, whose identification number would be saved in the update.

The following two pseudo-codes showing the cow object both in corda and Ethereum.



```
contract CowContract {
    enum STATE{ Good, Sick, Infectious, Dead }
    struct COW {
        uint   id;
        STATE state;
        string vet;
        uint   time;
    }
    mapping(address => COW[]) private cows;
```

Figure 3.14 : Cow definition in Solidity (Ethereum).

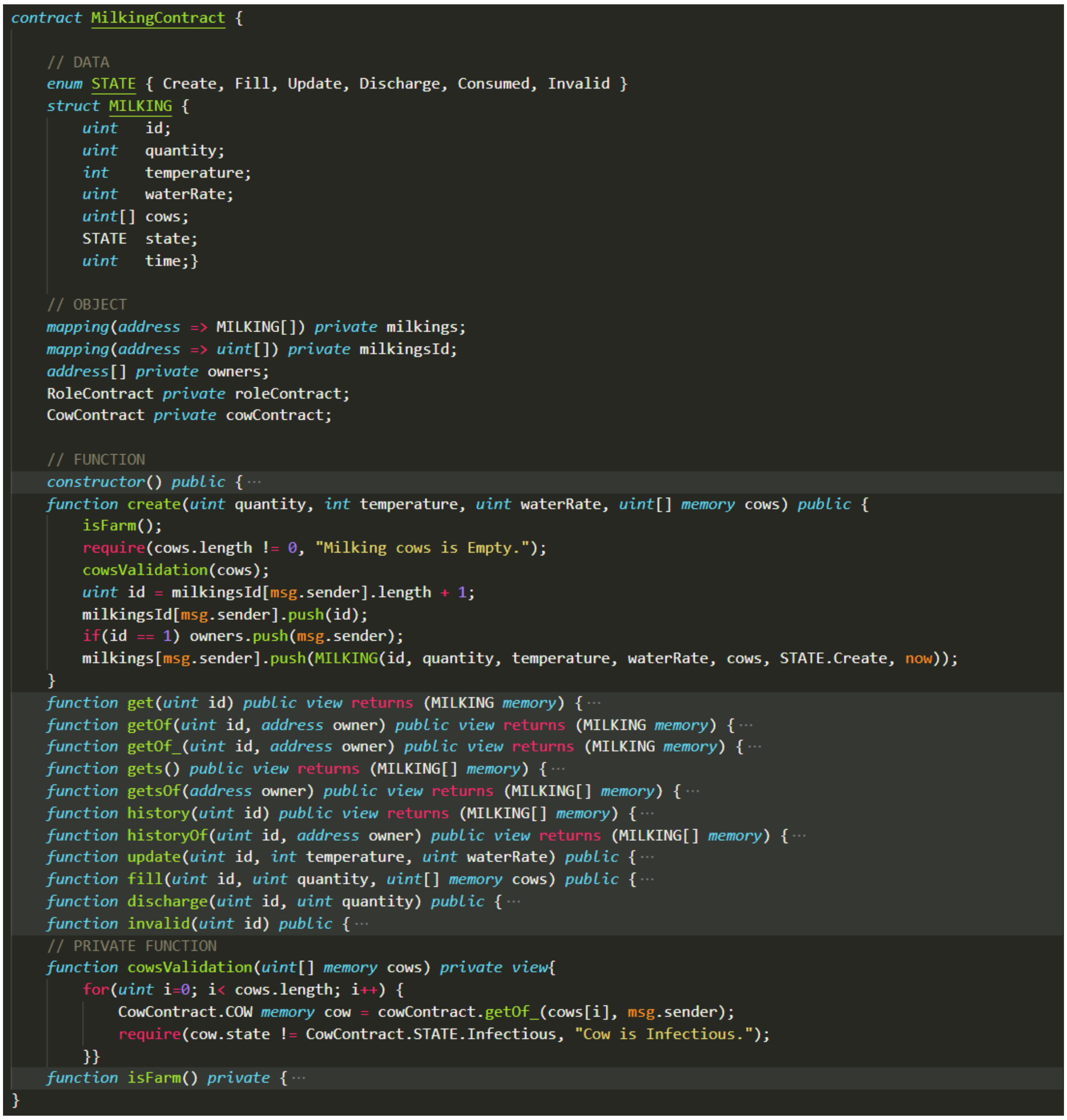


```
data class CowState(val id:Int,
                    val owner:Party,
                    val vet:String="",
                    val status:CowStatus=CowStatus.Good,
                    val time:LocalDateTime=time(),
                    override val participants:List<Party> =listOf()): ContractState {
```

Figure 3.15 : Cow State in Kotlin (Corda).

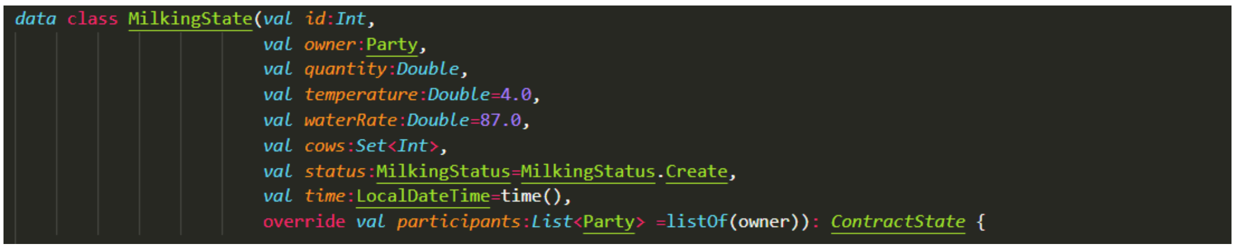
Once the form in which the cows are represented in the network is known, it is now possible to address the milk-creating nodes (milking) where the health of the cows is checked. before initialization of any milking process. This contract verifies whether one of the milked cows is infected with any infectious diseases according to their latest update, by checking the date of updates, determining the infection of one of the cows or through updating the status of the old cows which lead to the rejection of the whole milk and, therefore, breaking the network recording and the quantity of milk as well as dealing with it as invalid.

The following two pseudo-codes showing the milking contract both in corda and Ethereum.



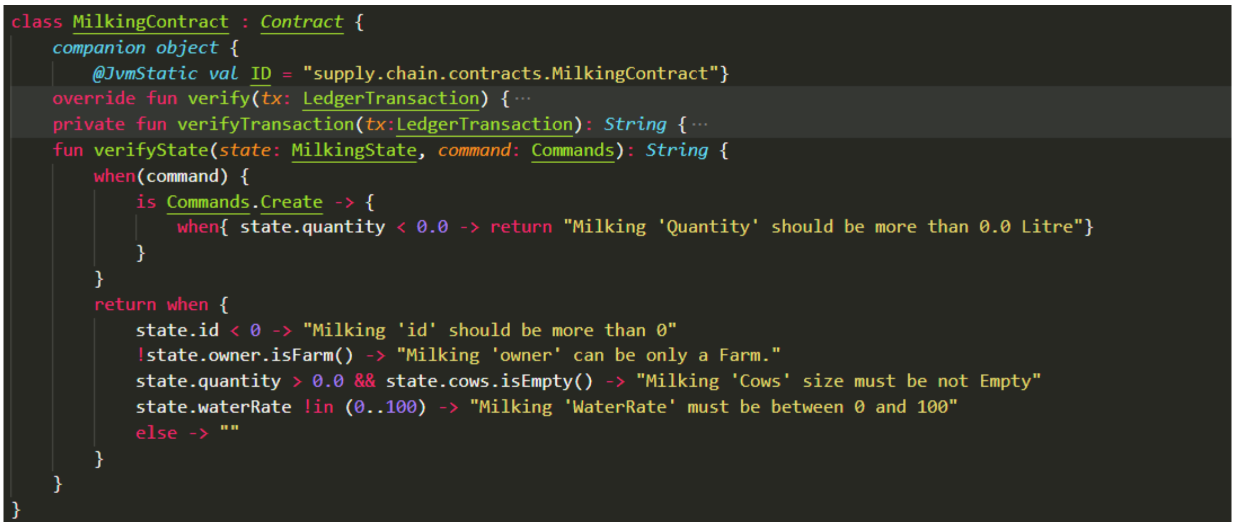
```solidity
contract MilkingContract {

    // DATA
    enum STATE { Create, Fill, Update, Discharge, Consumed, Invalid }
    struct MILKING {
        uint    id;
        uint    quantity;
        int     temperature;
        uint    waterRate;
        uint[] cows;
        STATE   state;
        uint    time;}

    // OBJECT
    mapping(address => MILKING[]) private milkings;
    mapping(address => uint[]) private milkingsId;
    address[] private owners;
    RoleContract private roleContract;
    CowContract private cowContract;

    // FUNCTION
    constructor() public {···
    function create(uint quantity, int temperature, uint waterRate, uint[] memory cows) public {
        isFarm();
        require(cows.length != 0, "Milking cows is Empty.");
        cowsValidation(cows);
        uint id = milkingsId[msg.sender].length + 1;
        milkingsId[msg.sender].push(id);
        if(id == 1) owners.push(msg.sender);
        milkings[msg.sender].push(MILKING(id, quantity, temperature, waterRate, cows, STATE.Create, now));
    }
    function get(uint id) public view returns (MILKING memory) {···
    function getOf(uint id, address owner) public view returns (MILKING memory) {···
    function getOf_(uint id, address owner) public view returns (MILKING memory) {···
    function gets() public view returns (MILKING[] memory) {···
    function getsOf(address owner) public view returns (MILKING[] memory) {···
    function history(uint id) public view returns (MILKING[] memory) {···
    function historyOf(uint id, address owner) public view returns (MILKING[] memory) {···
    function update(uint id, int temperature, uint waterRate) public {···
    function fill(uint id, uint quantity, uint[] memory cows) public {···
    function discharge(uint id, uint quantity) public {···
    function invalid(uint id) public {···
    // PRIVATE FUNCTION
    function cowsValidation(uint[] memory cows) private view{
        for(uint i=0; i< cows.length; i++) {
            CowContract.COW memory cow = cowContract.getOf_(cows[i], msg.sender);
            require(cow.state != CowContract.STATE.Infectious, "Cow is Infectious.");
        }}
    function isFarm() private {···
}
```

Figure 3.16 : Milking contract in Solidity (Ethereum).



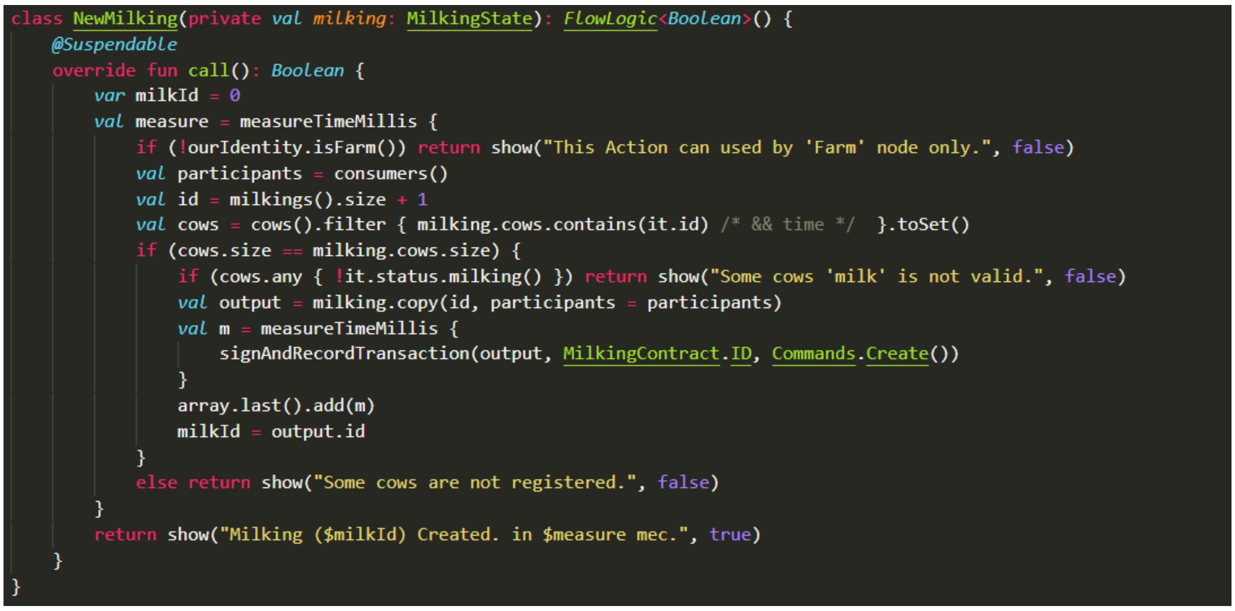
```kotlin
data class MilkingState(val id:Int,
                        val owner:Party,
                        val quantity:Double,
                        val temperature:Double=4.0,
                        val waterRate:Double=87.0,
                        val cows:Set<Int>,
                        val status:MilkingStatus=MilkingStatus.Create,
                        val time:LocalDateTime=time(),
                        override val participants:List<Party> =listOf(owner)): ContractState {
```

Figure 3.17 : Milking State in Kotlin (Corda).



```
class MilkingContract : Contract {
    companion object {
        @JvmStatic val ID = "supply.chain.contracts.MilkingContract"}
    override fun verify(tx: LedgerTransaction) {···
    private fun verifyTransaction(tx:LedgerTransaction): String {···
    fun verifyState(state: MilkingState, command: Commands): String {
        when(command) {
            is Commands.Create -> {
                when{ state.quantity < 0.0 -> return "Milking 'Quantity' should be more than 0.0 Litre"}
            }
        }
        return when {
            state.id < 0 -> "Milking 'id' should be more than 0"
            !state.owner.isFarm() -> "Milking 'owner' can be only a Farm."
            state.quantity > 0.0 && state.cows.isEmpty() -> "Milking 'Cows' size must be not Empty"
            state.waterRate !in (0..100) -> "Milking 'WaterRate' must be between 0 and 100"
            else -> ""
        }
    }
}
```

Figure 3.18 : Milking Contract in Kotlin (Corda).



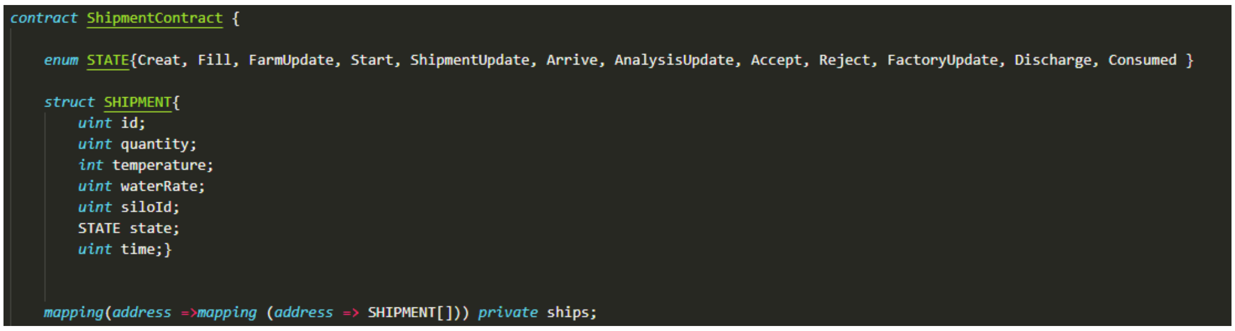
```
class NewMilking(private val milking: MilkingState): FlowLogic<Boolean>() {
    @Suspendable
    override fun call(): Boolean {
        var milkId = 0
        val measure = measureTimeMillis {
            if (!ourIdentity.isFarm()) return show("This Action can used by 'Farm' node only.", false)
            val participants = consumers()
            val id = milkings().size + 1
            val cows = cows().filter { milking.cows.contains(it.id) /* && time */  }.toSet()
            if (cows.size == milking.cows.size) {
                if (cows.any { !it.status.milking() }) return show("Some cows 'milk' is not valid.", false)
                val output = milking.copy(id, participants = participants)
                val m = measureTimeMillis {
                    signAndRecordTransaction(output, MilkingContract.ID, Commands.Create())
                }
                array.last().add(m)
                milkId = output.id
            }
            else return show("Some cows are not registered.", false)
        }
        return show("Milking ($milkId) Created. in $measure mec.", true)
    }
}
```

Figure 3.19 : Create new milking flow in Kotlin (Corda).

- **Shipment contracts (see contract No. 4 in Figure 3.13)**

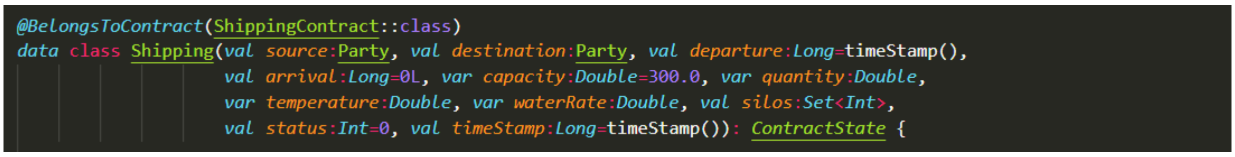
The importance of the IoT is obvious in the supply chain in rendering the system autonomous and the blockchain that achieves high transparency and data security.as soon as the shipment of milk from the farm arrives at the factory, after several contract at farm level to verify the health of the milk and subject it to surveillance all the way to the factory by use of sensors linked to the network (***Figure 3.4***) in order to avoid any violations.

The following two pseudo-codes showing the shipment object both in corda and Ethereum.



```
contract ShipmentContract {

    enum STATE{Creat, Fill, FarmUpdate, Start, ShipmentUpdate, Arrive, AnalysisUpdate, Accept, Reject, FactoryUpdate, Discharge, Consumed }

    struct SHIPMENT{
        uint id;
        uint quantity;
        int temperature;
        uint waterRate;
        uint siloId;
        STATE state;
        uint time;}

    mapping(address =>mapping (address => SHIPMENT[])) private ships;
```

Figure 3.20 : Shipment definition in Solidity (Ethereum).



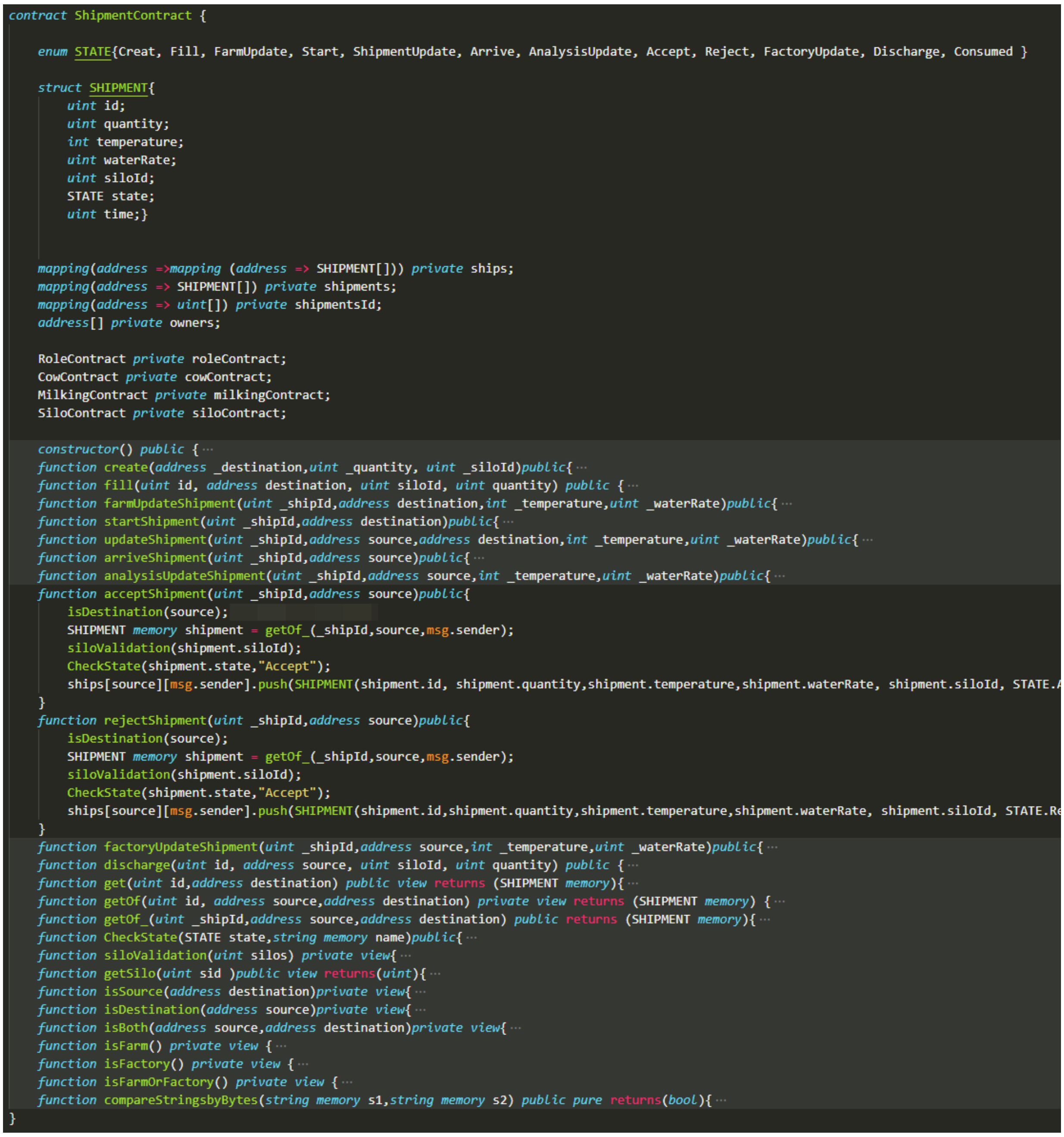
```
@BelongsToContract(ShippingContract::class)
data class Shipping(val source:Party, val destination:Party, val departure:Long=timeStamp(),
                    val arrival:Long=0L, var capacity:Double=300.0, var quantity:Double,
                    var temperature:Double, var waterRate:Double, val silos:Set<Int>,
                    val status:Int=0, val timeStamp:Long=timeStamp()): ContractState {
```

Figure 3.21 : Shipment State in Kotlin (Corda).

And as a final step after the shipment arrives, Milk is tested for quality at the factory laboratories, where the result is either accept or reject the shipment. (The result is presented in "***status***" value)

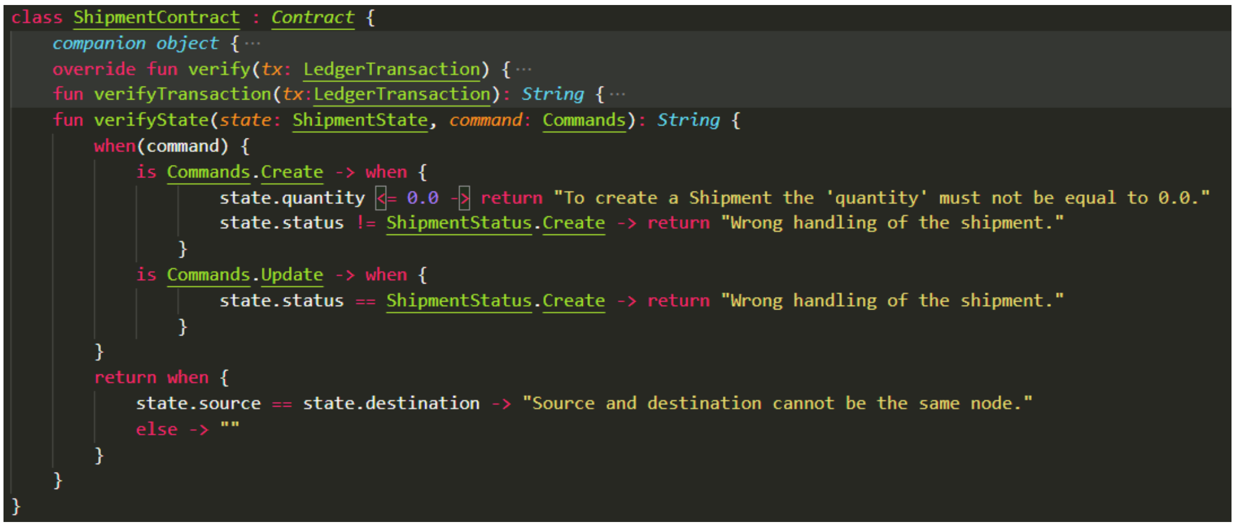
- **The shipment is accepted (ShipmentStatus.Accept) :** To accepting the shipment it must meets all the standards, is set as an acceptable shipment and the network automatically triggers the contract complete ***payment***, which in turn determines the amount shipped by the agreed price when the partnership between the two parties "***farm and factory***" was created (***partnership.price***). The amount of money previously reserved during the establishment of the shipment is subtracted from the account of the consumer (***partnership -> reserved_balance***).
- **The shipment is rejected (ShipmentStatus.Reject) :** In the event of rejection of the shipment. If the shipment doesn't meet the health standards, is set as a rejected shipment and the network automatically triggers the contract rejecting the deal, which in turn evaluates the amount shipped at the agreed price upon the partnership formed. The amount is re-released customer account that was previously reserved during the shipment creation (***partnership -> reserved_balance***).

The following pseudo-codes showing this contract with the accept and reject part on both Corda and Ethereum.



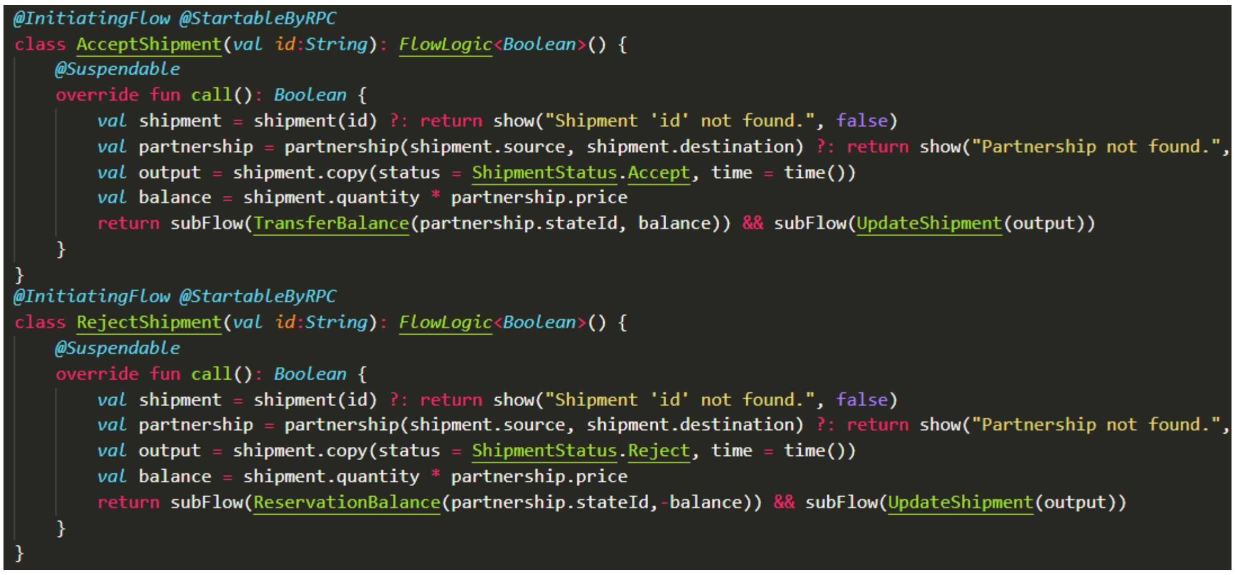
```
contract ShipmentContract {

    enum STATE{Creat, Fill, FarmUpdate, Start, ShipmentUpdate, Arrive, AnalysisUpdate, Accept, Reject, FactoryUpdate, Discharge, Consumed }

    struct SHIPMENT{
        uint id;
        uint quantity;
        int temperature;
        uint waterRate;
        uint siloId;
        STATE state;
        uint time;}


    mapping(address =>mapping (address => SHIPMENT[])) private ships;
    mapping(address => SHIPMENT[]) private shipments;
    mapping(address => uint[]) private shipmentsId;
    address[] private owners;

    RoleContract private roleContract;
    CowContract private cowContract;
    MilkingContract private milkingContract;
    SiloContract private siloContract;

    constructor() public {···
    function create(address _destination,uint _quantity, uint _siloId)public{···
    function fill(uint id, address destination, uint siloId, uint quantity) public {···
    function farmUpdateShipment(uint _shipId,address destination,int _temperature,uint _waterRate)public{···
    function startShipment(uint _shipId,address destination)public{···
    function updateShipment(uint _shipId,address source,address destination,int _temperature,uint _waterRate)public{···
    function arriveShipment(uint _shipId,address source)public{···
    function analysisUpdateShipment(uint _shipId,address source,int _temperature,uint _waterRate)public{···
    function acceptShipment(uint _shipId,address source)public{
        isDestination(source);
        SHIPMENT memory shipment = getOf_(_shipId,source,msg.sender);
        siloValidation(shipment.siloId);
        CheckState(shipment.state,"Accept");
        ships[source][msg.sender].push(SHIPMENT(shipment.id, shipment.quantity,shipment.temperature,shipment.waterRate, shipment.siloId, STATE.A
    }
    function rejectShipment(uint _shipId,address source)public{
        isDestination(source);
        SHIPMENT memory shipment = getOf_(_shipId,source,msg.sender);
        siloValidation(shipment.siloId);
        CheckState(shipment.state,"Accept");
        ships[source][msg.sender].push(SHIPMENT(shipment.id,shipment.quantity,shipment.temperature,shipment.waterRate, shipment.siloId, STATE.Re
    }
    function factoryUpdateShipment(uint _shipId,address source,int _temperature,uint _waterRate)public{···
    function discharge(uint id, address source, uint siloId, uint quantity) public {···
    function get(uint id,address destination) public view returns (SHIPMENT memory){···
    function getOf(uint id, address source,address destination) private view returns (SHIPMENT memory) {···
    function getOf_(uint _shipId,address source,address destination) public returns (SHIPMENT memory){···
    function CheckState(STATE state,string memory name)public{···
    function siloValidation(uint silos) private view{···
    function getSilo(uint sid )public view returns(uint){···
    function isSource(address destination)private view{···
    function isDestination(address source)private view{···
    function isBoth(address source,address destination)private view{···
    function isFarm() private view {···
    function isFactory() private view {···
    function isFarmOrFactory() private view {···
    function compareStringsbyBytes(string memory s1,string memory s2) public pure returns(bool){···
}
```

Figure 3.22 : Shipment smart contract in Solidity (Ethereum).



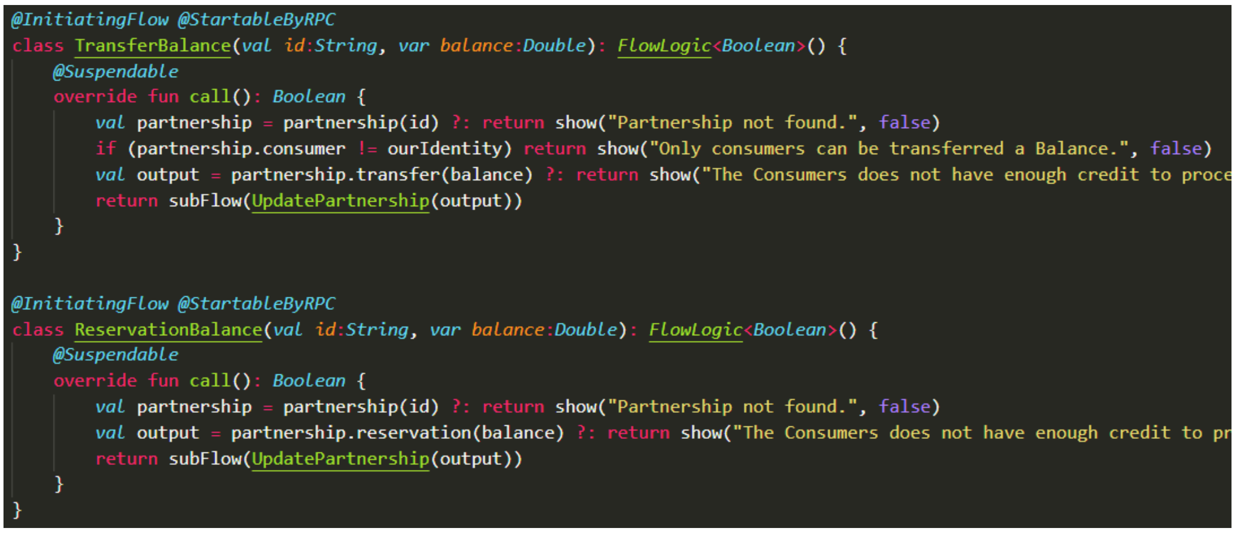
```
class ShipmentContract : Contract {
    companion object {···
    override fun verify(tx: LedgerTransaction) {···
    fun verifyTransaction(tx:LedgerTransaction): String {···
    fun verifyState(state: ShipmentState, command: Commands): String {
        when(command) {
            is Commands.Create -> when {
                    state.quantity <= 0.0 -> return "To create a Shipment the 'quantity' must not be equal to 0.0."
                    state.status != ShipmentStatus.Create -> return "Wrong handling of the shipment."
                }
            is Commands.Update -> when {
                    state.status == ShipmentStatus.Create -> return "Wrong handling of the shipment."
                }
        }
        return when {
            state.source == state.destination -> "Source and destination cannot be the same node."
            else -> ""
        }
    }
}
```

Figure 3.23 : Shipment contract in Kotlin (Corda).



```
@InitiatingFlow @StartableByRPC
class AcceptShipment(val id:String): FlowLogic<Boolean>() {
    @Suspendable
    override fun call(): Boolean {
        val shipment = shipment(id) ?: return show("Shipment 'id' not found.", false)
        val partnership = partnership(shipment.source, shipment.destination) ?: return show("Partnership not found.",
        val output = shipment.copy(status = ShipmentStatus.Accept, time = time())
        val balance = shipment.quantity * partnership.price
        return subFlow(TransferBalance(partnership.stateId, balance)) && subFlow(UpdateShipment(output))
    }
}
@InitiatingFlow @StartableByRPC
class RejectShipment(val id:String): FlowLogic<Boolean>() {
    @Suspendable
    override fun call(): Boolean {
        val shipment = shipment(id) ?: return show("Shipment 'id' not found.", false)
        val partnership = partnership(shipment.source, shipment.destination) ?: return show("Partnership not found.",
        val output = shipment.copy(status = ShipmentStatus.Reject, time = time())
        val balance = shipment.quantity * partnership.price
        return subFlow(ReservationBalance(partnership.stateId,-balance)) && subFlow(UpdateShipment(output))
    }
}
```

Figure 3.24 : Accept and Reject Shipment flow in Kotlin (Corda).

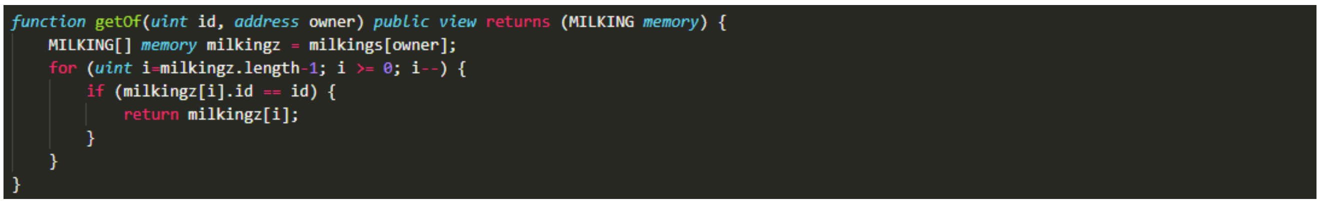


```
@InitiatingFlow @StartableByRPC
class TransferBalance(val id:String, var balance:Double): FlowLogic<Boolean>() {
    @Suspendable
    override fun call(): Boolean {
        val partnership = partnership(id) ?: return show("Partnership not found.", false)
        if (partnership.consumer != ourIdentity) return show("Only consumers can be transferred a Balance.", false)
        val output = partnership.transfer(balance) ?: return show("The Consumers does not have enough credit to proce
        return subFlow(UpdatePartnership(output))
    }
}

@InitiatingFlow @StartableByRPC
class ReservationBalance(val id:String, var balance:Double): FlowLogic<Boolean>() {
    @Suspendable
    override fun call(): Boolean {
        val partnership = partnership(id) ?: return show("Partnership not found.", false)
        val output = partnership.reservation(balance) ?: return show("The Consumers does not have enough credit to pr
        return subFlow(UpdatePartnership(output))
    }
}
```

Figure 3.25 : Transfer and Reservation Balance flow in Kotlin (Corda).

4.6.2. Search operation

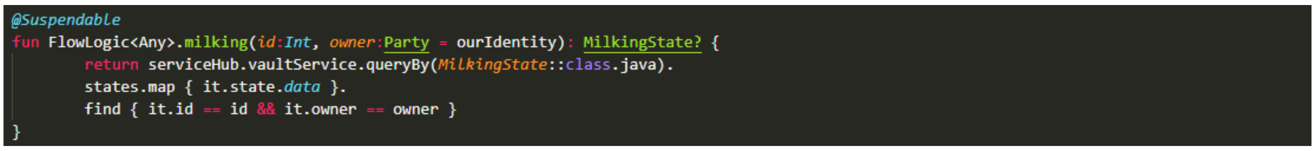
Any user in the system could perform a research operation in the blockchain, the user needs to introduce all required parameters in the message request, For example, one can apply a research about milking by introducing the identifier of milking, this operation will return all the information related to milking object (id,milk quantity,milk temperature,milk water rate, list of milked cows,milk state, time of milking).
The following pseudo-codes shows the search operation in milking objects both in Ethereum and Corda :



```
function getOf(uint id, address owner) public view returns (MILKING memory) {
    MILKING[] memory milkingz = milkings[owner];
    for (uint i=milkingz.length-1; i >= 0; i--) {
        if (milkingz[i].id == id) {
            return milkingz[i];
        }
    }
}
```

Figure 3.26 : Getting Milking last state (Ethereum).

In Ethereum we used a loop that begins from the last object of state list(list of milking objects), so we can get the last state of the wanted object by verifying the id and the owner.



```
@Suspendable
fun FlowLogic<Any>.milking(id:Int, owner:Party = ourIdentity): MilkingState? {
        return serviceHub.vaultService.queryBy(MilkingState::class.java).
        states.map { it.state.data }.
        find { it.id == id && it.owner == owner }
}
```

Figure 3.27 : Getting Milking last state (Corda).

Corda unlike Ethereum no need for a loop, it already exist a state of objects, we just need to verify the id and the owner and get the latest state of the required object.

5. Run and application test

Within the authors implementation the network is interactable via command line interface (***CLI***). The front-end and api layer are not implemented for the application since the author considers them to be outside of the scope and they do not differ from average web application.

5.1. Application running and test in Ethereum

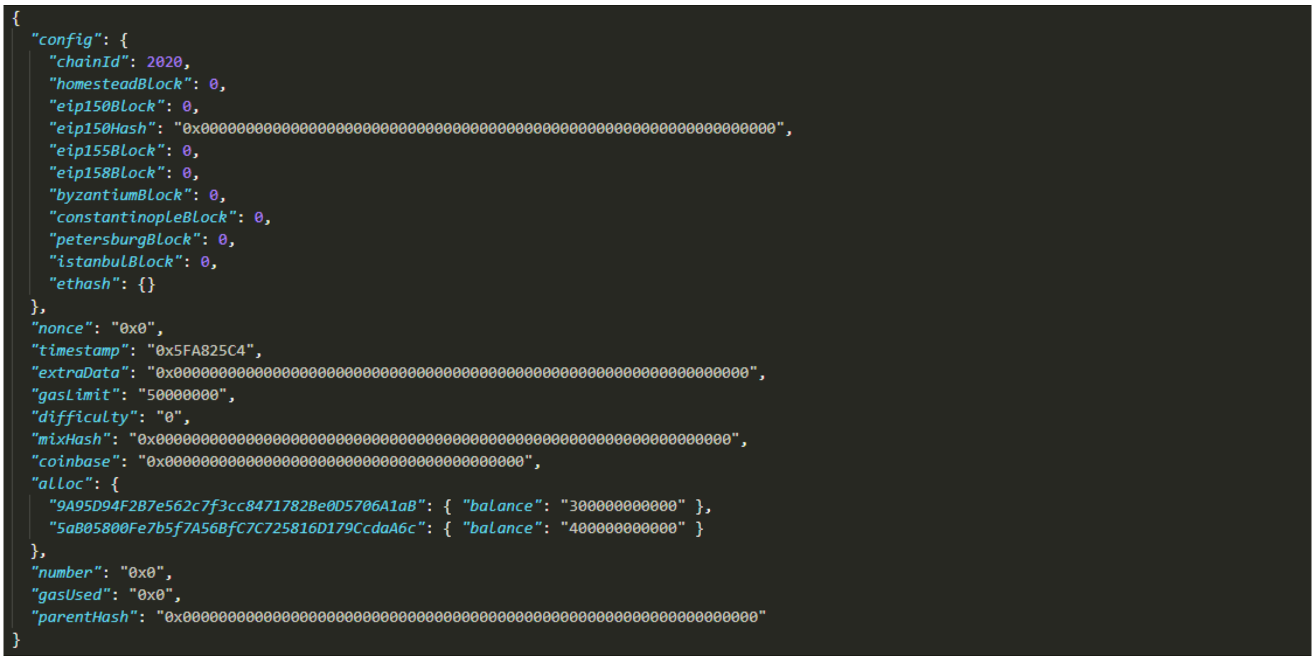
5.1.1. Deploy nodes

First of all we start by installing **Geth**, then create a directory for the nodes, after that we start by going to the directory of node and create an account (optional) with this command

```
geth --datadir ./data account new
```

After creating the account we initial the genesis which is named **"*first*"** and has ***json*** extension of the blockchain with a command:

```
geth --datadir ./data init ../first.json
```



```
{
  "config": {
    "chainId": 2020,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "ethash": {}
  },
  "nonce": "0x0",
  "timestamp": "0x5FA825C4",
  "extraData": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "gasLimit": "50000000",
  "difficulty": "0",
  "mixHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x0000000000000000000000000000000000000000",
  "alloc": {
    "9A95D94F2B7e562c7f3cc8471782Be0D5706A1aB": { "balance": "300000000000" },
    "5aB05800Fe7b5f7A56BfC7C725816D179CcdaA6c": { "balance": "400000000000" }
  },
  "number": "0x0",
  "gasUsed": "0x0",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000"
}
```

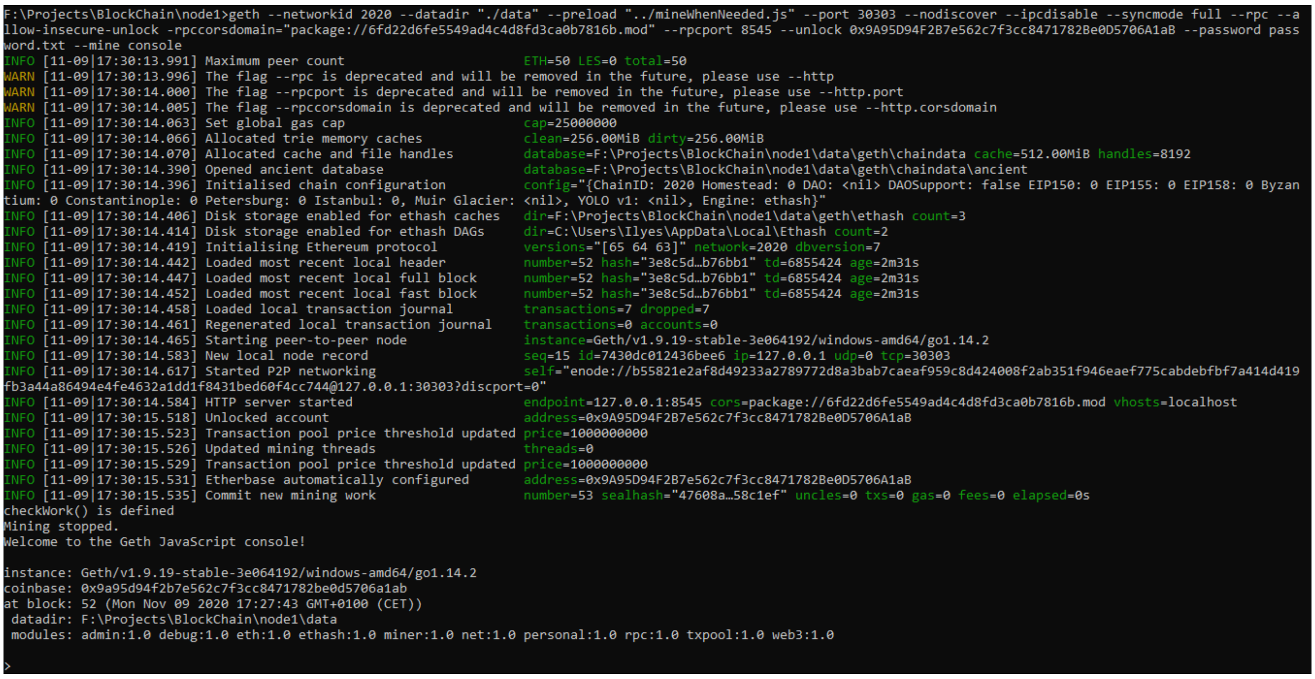
Figure 3.28 : Configuration of genesis block (first.json).

5.1.2. Run nodes

We start our node by a typing this command:

```
geth --networkid (INT) --datadir "./data" --port (PORT) --rpc -rpccorsdomain = "package://6fd22d6fe5549ad4c4d8fd3ca0b7816b.mod" --rpcport (PRC-PORT) ---mine console
```

and this how node will be looking after the execution of the command in cmd



```
F:\Projects\BlockChain\node1>geth --networkid 2020 --datadir "./data" --preload "../mineWhenNeeded.js" --port 30303 --nodiscover --ipcdisable --syncmode full --rpc --a
llow-insecure-unlock -rpccorsdomain="package://6fd22d6fe5549ad4c4d8fd3ca0b7816b.mod" --rpcport 8545 --unlock 0x9A95D94F2B7e562c7f3cc8471782Be0D5706A1aB --password pass
word.txt --mine console
INFO [11-09|17:30:13.991] Maximum peer count                       ETH=50 LES=0 total=50
WARN [11-09|17:30:13.996] The flag --rpc is deprecated and will be removed in the future, please use --http
WARN [11-09|17:30:14.000] The flag --rpcport is deprecated and will be removed in the future, please use --http.port
WARN [11-09|17:30:14.005] The flag --rpccorsdomain is deprecated and will be removed in the future, please use --http.corsdomain
INFO [11-09|17:30:14.063] Set global gas cap                       cap=25000000
INFO [11-09|17:30:14.066] Allocated trie memory caches             clean=256.00MiB dirty=256.00MiB
INFO [11-09|17:30:14.070] Allocated cache and file handles         database=F:\Projects\BlockChain\node1\data\geth\chaindata cache=512.00MiB handles=8192
INFO [11-09|17:30:14.390] Opened ancient database                  database=F:\Projects\BlockChain\node1\data\geth\chaindata\ancient
INFO [11-09|17:30:14.396] Initialised chain configuration          config="{ChainID: 2020 Homestead: 0 DAO: <nil> DAOSupport: false EIP150: 0 EIP155: 0 EIP158: 0 Byzan
tium: 0 Constantinople: 0 Petersburg: 0 Istanbul: 0, Muir Glacier: <nil>, YOLO v1: <nil>, Engine: ethash}"
INFO [11-09|17:30:14.406] Disk storage enabled for ethash caches   dir=F:\Projects\BlockChain\node1\data\geth\ethash count=3
INFO [11-09|17:30:14.414] Disk storage enabled for ethash DAGs     dir=C:\Users\Ilyes\AppData\Local\Ethash count=2
INFO [11-09|17:30:14.419] Initialising Ethereum protocol           versions="[65 64 63]" network=2020 dbversion=7
INFO [11-09|17:30:14.442] Loaded most recent local header          number=52 hash="3e8c5d…b76bb1" td=6855424 age=2m31s
INFO [11-09|17:30:14.447] Loaded most recent local full block      number=52 hash="3e8c5d…b76bb1" td=6855424 age=2m31s
INFO [11-09|17:30:14.452] Loaded most recent local fast block      number=52 hash="3e8c5d…b76bb1" td=6855424 age=2m31s
INFO [11-09|17:30:14.458] Loaded local transaction journal         transactions=7 dropped=7
INFO [11-09|17:30:14.461] Regenerated local transaction journal    transactions=0 accounts=0
INFO [11-09|17:30:14.465] Starting peer-to-peer node               instance=Geth/v1.9.19-stable-3e064192/windows-amd64/go1.14.2
INFO [11-09|17:30:14.583] New local node record                    seq=15 id=7430dc012436bee6 ip=127.0.0.1 udp=0 tcp=30303
INFO [11-09|17:30:14.617] Started P2P networking                   self="enode://b55821e2af8d49233a2789772d8a3bab7caeaf959c8d424008f2ab351f946eaef775cabdebfbf7a414d419
fb3a44a86494e4fe4632a1dd1f8431bed60f4cc744@127.0.0.1:30303?discport=0"
INFO [11-09|17:30:14.584] HTTP server started                      endpoint=127.0.0.1:8545 cors=package://6fd22d6fe5549ad4c4d8fd3ca0b7816b.mod vhosts=localhost
INFO [11-09|17:30:15.518] Unlocked account                         address=0x9A95D94F2B7e562c7f3cc8471782Be0D5706A1aB
INFO [11-09|17:30:15.523] Transaction pool price threshold updated price=1000000000
INFO [11-09|17:30:15.526] Updated mining threads                   threads=0
INFO [11-09|17:30:15.529] Transaction pool price threshold updated price=1000000000
INFO [11-09|17:30:15.531] Etherbase automatically configured       address=0x9A95D94F2B7e562c7f3cc8471782Be0D5706A1aB
INFO [11-09|17:30:15.535] Commit new mining work                   number=53 sealhash="47608a…58c1ef" uncles=0 txs=0 gas=0 fees=0 elapsed=0s
checkWork() is defined
Mining stopped.
Welcome to the Geth JavaScript console!

instance: Geth/v1.9.19-stable-3e064192/windows-amd64/go1.14.2
coinbase: 0x9a95d94f2b7e562c7f3cc8471782be0d5706a1ab
at block: 52 (Mon Nov 09 2020 17:27:43 GMT+0100 (CET))
 datadir: F:\Projects\BlockChain\node1\data
 modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

>
```

Figure 3.29 : Ethereum node in CLI.

5.1.3. Connect nodes

After starting nodes we can manual add peers by typing the following, and let us choose the network structure and no need for node to connect with each other and no need fully connected node:

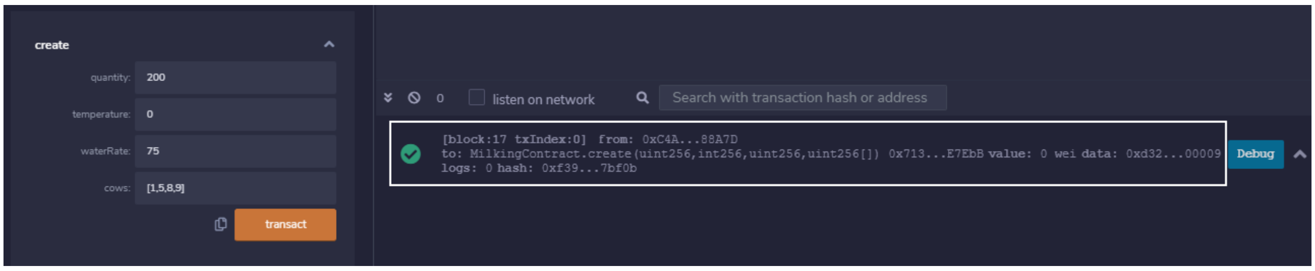
```
admin.addPeer("Argument")
```

Argument representing the output of the next command in the second node:

```
admin.nodeInfo.enode
```

5.1.4. Execute Contract

To speed up the development process and without the need to create an interface via web technologies, we took advantage of the ***Remix IDE*** interface as shown in ***Figure 3.30***



create

quantity: 200

temperature: 0

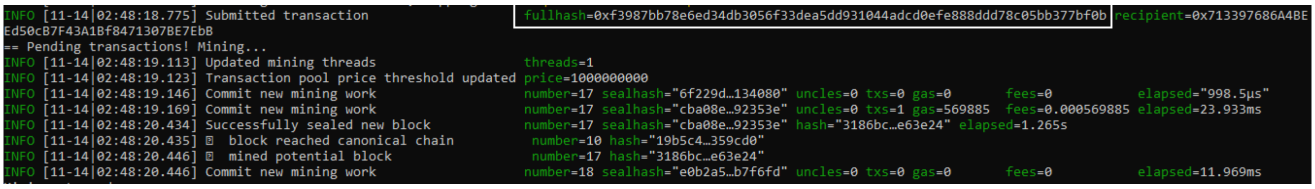
waterRate: 75

cows: [1,5,8,9]

```
[block:17 txIndex:0]  from: 0xC4A...88A7D
to: MilkingContract.create(uint256,int256,uint256,uint256[]) 0x713...E7EbB value: 0 wei data: 0xd32...00009
logs: 0 hash: 0xf39...7bf0b
```

Figure 3.30 : Add new milking and completed transaction notification on remix.

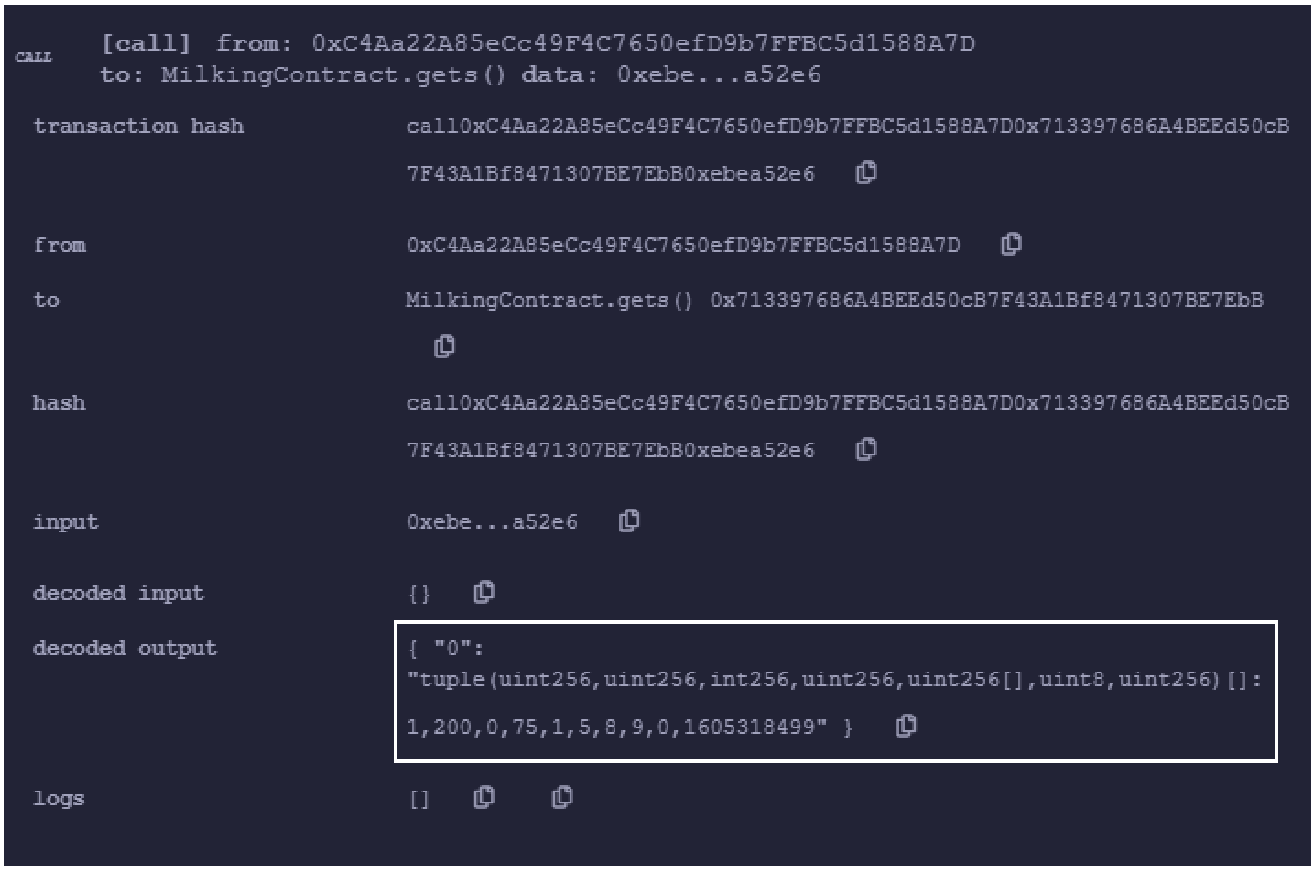
On the other hand, the addition of the new block containing the previous transaction can be seen on the ***Geth CLI*** interface as shown in the following figure



```
INFO [11-14|02:48:18.775] Submitted transaction                    fullhash=0xf3987bb78e6ed34db3056f33dea5dd931044adcd0efe888ddd78c05bb377bf0b recipient=0x713397686A4BE
Ed50cB7F43A1Bf8471307BE7EbB
== Pending transactions! Mining...
INFO [11-14|02:48:19.113] Updated mining threads                   threads=1
INFO [11-14|02:48:19.123] Transaction pool price threshold updated price=1000000000
INFO [11-14|02:48:19.146] Commit new mining work                   number=17 sealhash="6f229d…134080" uncles=0 txs=0 gas=0       fees=0           elapsed="998.5µs"
INFO [11-14|02:48:19.169] Commit new mining work                   number=17 sealhash="cba08e…92353e" uncles=0 txs=1 gas=569885  fees=0.000569885 elapsed=23.933ms
INFO [11-14|02:48:20.434] Successfully sealed new block            number=17 sealhash="cba08e…92353e" hash="3186bc…e63e24" elapsed=1.265s
INFO [11-14|02:48:20.435] ⍰  block reached canonical chain          number=10 hash="19b5c4…359cd0"
INFO [11-14|02:48:20.446] ⍰  mined potential block                  number=17 hash="3186bc…e63e24"
INFO [11-14|02:48:20.446] Commit new mining work                   number=18 sealhash="e0b2a5…b7f6fd" uncles=0 txs=0 gas=0       fees=0           elapsed=11.969ms
```

Figure 3.31 : Add and mine, milking transaction (Block).

The content of the transaction can be known via the remix console, as shown in the following figure



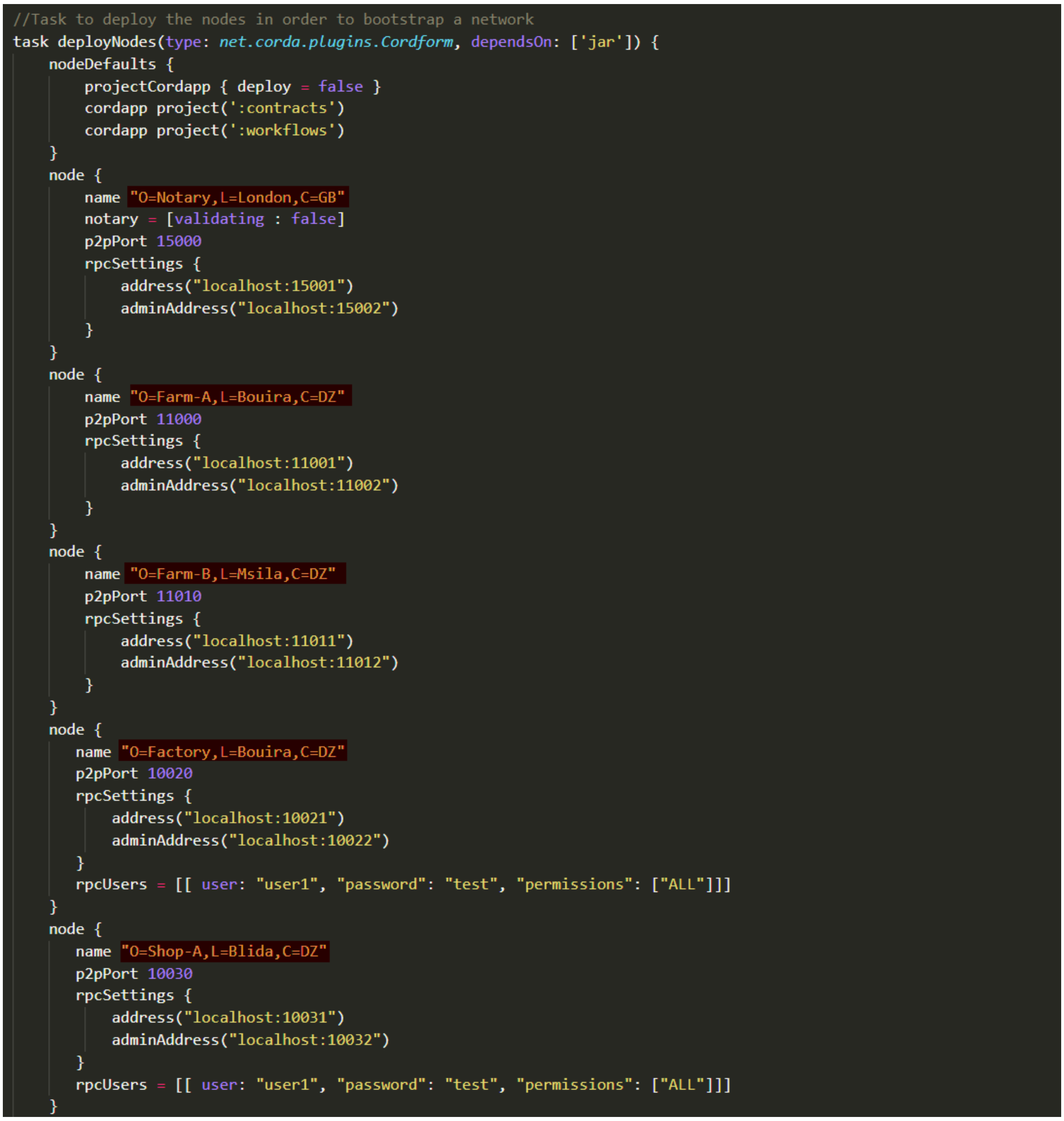
```
CALL   [call] from: 0xC4Aa22A85eCc49F4C7650efD9b7FFBC5d1588A7D
       to: MilkingContract.gets() data: 0xebe...a52e6

 transaction hash        call0xC4Aa22A85eCc49F4C7650efD9b7FFBC5d1588A7D0x713397686A4BEEd50cB
                         7F43A1Bf8471307BE7EbB0xebea52e6

 from                    0xC4Aa22A85eCc49F4C7650efD9b7FFBC5d1588A7D

 to                      MilkingContract.gets() 0x713397686A4BEEd50cB7F43A1Bf8471307BE7EbB

 hash                    call0xC4Aa22A85eCc49F4C7650efD9b7FFBC5d1588A7D0x713397686A4BEEd50cB
                         7F43A1Bf8471307BE7EbB0xebea52e6

 input                   0xebe...a52e6

 decoded input           {}

 decoded output          { "0":
                         "tuple(uint256,uint256,int256,uint256,uint256[],uint8,uint256)[]:
                         1,200,0,75,1,5,8,9,0,1605318499" }

 logs                    []
```

Figure 3.32 : Show milking transaction on remix console.

5.2. Application running and test in Corda

5.2.1. Deploy nodes

The ***deployNodes*** task in the build.grade file is edited to represent the current use case.



```
//Task to deploy the nodes in order to bootstrap a network
task deployNodes(type: net.corda.plugins.Cordform, dependsOn: ['jar']) {
    nodeDefaults {
        projectCordapp { deploy = false }
        cordapp project(':contracts')
        cordapp project(':workflows')
    }
    node {
        name "O=Notary,L=London,C=GB"
        notary = [validating : false]
        p2pPort 15000
        rpcSettings {
            address("localhost:15001")
            adminAddress("localhost:15002")
        }
    }
    node {
        name "O=Farm-A,L=Bouira,C=DZ"
        p2pPort 11000
        rpcSettings {
            address("localhost:11001")
            adminAddress("localhost:11002")
        }
    }
    node {
        name "O=Farm-B,L=Msila,C=DZ"
        p2pPort 11010
        rpcSettings {
            address("localhost:11011")
            adminAddress("localhost:11012")
        }
    }
    node {
       name "O=Factory,L=Bouira,C=DZ"
       p2pPort 10020
       rpcSettings {
           address("localhost:10021")
           adminAddress("localhost:10022")
       }
       rpcUsers = [[ user: "user1", "password": "test", "permissions": ["ALL"]]]
    }
    node {
       name "O=Shop-A,L=Blida,C=DZ"
       p2pPort 10030
       rpcSettings {
           address("localhost:10031")
           adminAddress("localhost:10032")
       }
       rpcUsers = [[ user: "user1", "password": "test", "permissions": ["ALL"]]]
    }
```

Figure 3.33 : DeployNodes task in build.gradle file.

Next we need can run it in our terminal window with the gradle command

```
./gradlew clean deployNodes
```

This is creates the build catalogues to the structure, with that all the kotlin code is compiled as well.

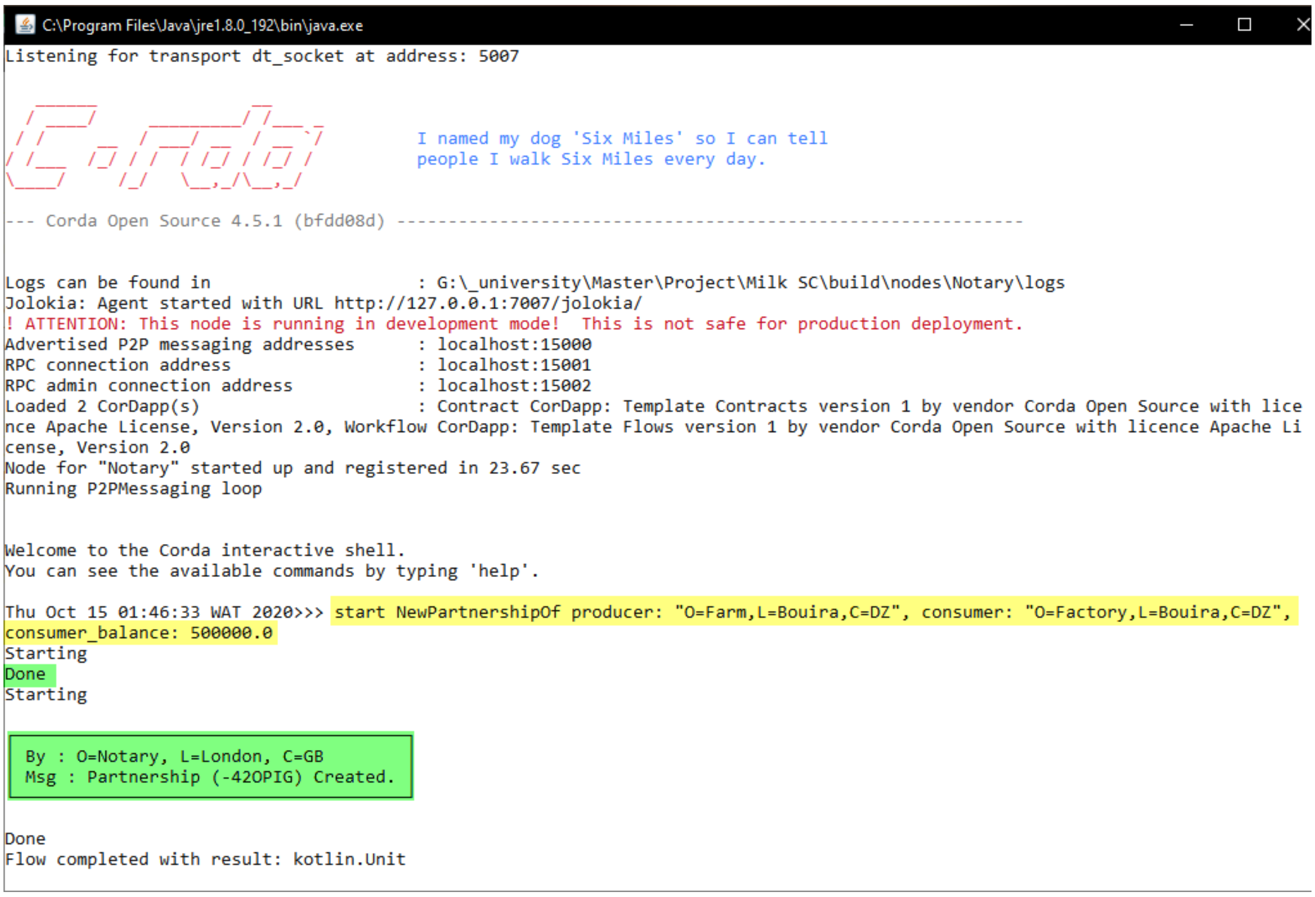
5.2.2. Run nodes and execute contract

To start all the nodes a second command is necessary to run in the same root folder

```
build/nodes/runnodes
```

To create a new partnership between 2 node we need to run this command

```
start NewPartnership producer: "node1 name", consumer: "node2 name", consumer_balance: Double
```

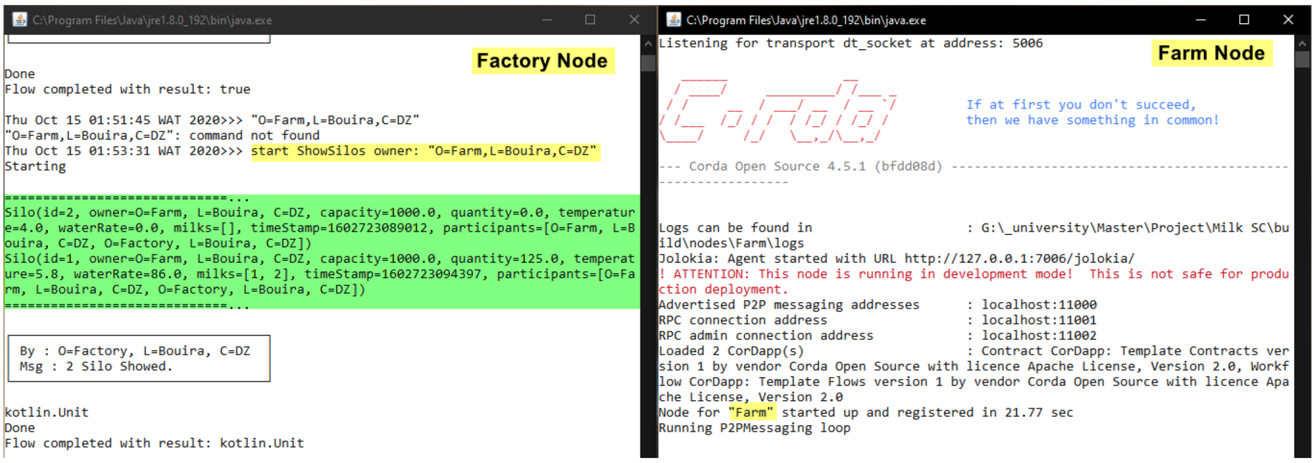


```
C:\Program Files\Java\jre1.8.0_192\bin\java.exe
Listening for transport dt_socket at address: 5007

   ______               __
  / ____/     _________/ /___ _
 / /     __  / ___/ __  / __ `/         I named my dog 'Six Miles' so I can tell
/ /___  /_/ / /  / /_/ / /_/ /          people I walk Six Miles every day.
\____/     /_/   \__,_/\__,_/

--- Corda Open Source 4.5.1 (bfdd08d) -------------------------------------------------------------


Logs can be found in                    : G:\_university\Master\Project\Milk SC\build\nodes\Notary\logs
Jolokia: Agent started with URL http://127.0.0.1:7007/jolokia/
! ATTENTION: This node is running in development mode!  This is not safe for production deployment.
Advertised P2P messaging addresses      : localhost:15000
RPC connection address                  : localhost:15001
RPC admin connection address            : localhost:15002
Loaded 2 CorDapp(s)                     : Contract CorDapp: Template Contracts version 1 by vendor Corda Open Source with lice
nce Apache License, Version 2.0, Workflow CorDapp: Template Flows version 1 by vendor Corda Open Source with licence Apache Li
cense, Version 2.0
Node for "Notary" started up and registered in 23.67 sec
Running P2PMessaging loop


Welcome to the Corda interactive shell.
You can see the available commands by typing 'help'.

Thu Oct 15 01:46:33 WAT 2020>>> start NewPartnershipOf producer: "O=Farm,L=Bouira,C=DZ", consumer: "O=Factory,L=Bouira,C=DZ",
consumer_balance: 500000.0
Starting
Done
Starting

  By : O=Notary, L=London, C=GB
  Msg : Partnership (-420PIG) Created.

Done
Flow completed with result: kotlin.Unit
```

Figure 3.34 : Notary node CLI window create new partnership.

To see the silos information of one of the parties from the partnership, it is sufficient to run the following command

```
start ShowSilos owner: "node name"
```



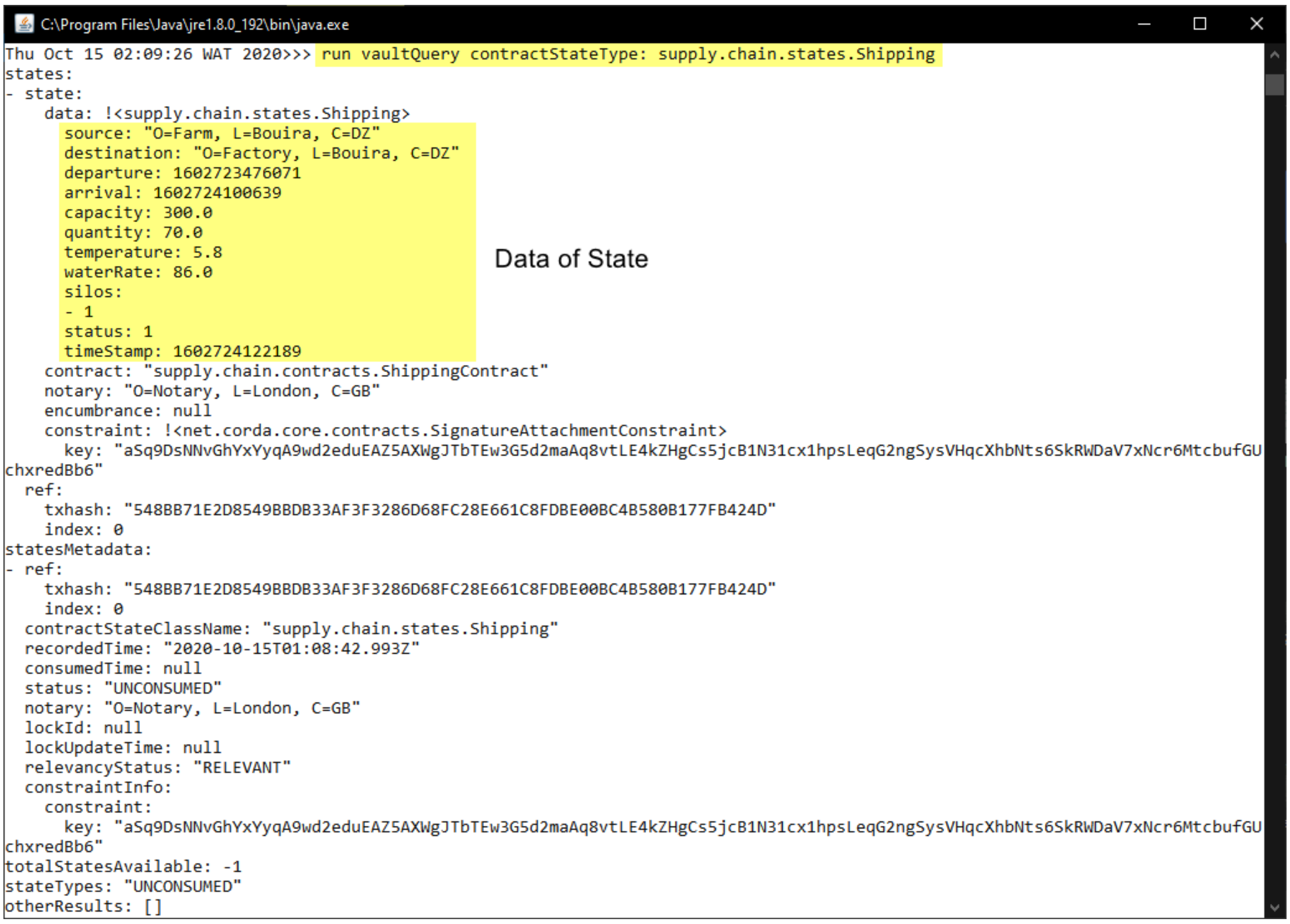
```
Done
Flow completed with result: true

Thu Oct 15 01:51:45 WAT 2020>>> "O=Farm,L=Bouira,C=DZ"
"O=Farm,L=Bouira,C=DZ": command not found
Thu Oct 15 01:53:31 WAT 2020>>> start ShowSilos owner: "O=Farm,L=Bouira,C=DZ"
Starting

============================...
Silo(id=2, owner=O=Farm, L=Bouira, C=DZ, capacity=1000.0, quantity=0.0, temperatur
e=4.0, waterRate=0.0, milks=[], timeStamp=1602723089012, participants=[O=Farm, L=B
ouira, C=DZ, O=Factory, L=Bouira, C=DZ])
Silo(id=1, owner=O=Farm, L=Bouira, C=DZ, capacity=1000.0, quantity=125.0, temperat
ure=5.8, waterRate=86.0, milks=[1, 2], timeStamp=1602723094397, participants=[O=Fa
rm, L=Bouira, C=DZ, O=Factory, L=Bouira, C=DZ])
============================...

 By : O=Factory, L=Bouira, C=DZ
 Msg : 2 Silo Showed.

kotlin.Unit
Done
Flow completed with result: kotlin.Unit
```

```
Listening for transport dt_socket at address: 5006

                                    If at first you don't succeed,
                                    then we have something in common!

--- Corda Open Source 4.5.1 (bfdd08d) ------------------------------------------
-----------------

Logs can be found in                    : G:\_university\Master\Project\Milk SC\bu
ild\nodes\Farm\logs
Jolokia: Agent started with URL http://127.0.0.1:7006/jolokia/
! ATTENTION: This node is running in development mode!  This is not safe for produ
ction deployment.
Advertised P2P messaging addresses      : localhost:11000
RPC connection address                  : localhost:11001
RPC admin connection address            : localhost:11002
Loaded 2 CorDapp(s)                     : Contract CorDapp: Template Contracts ver
sion 1 by vendor Corda Open Source with licence Apache License, Version 2.0, Workf
low CorDapp: Template Flows version 1 by vendor Corda Open Source with licence Apa
che License, Version 2.0
Node for "Farm" started up and registered in 21.77 sec
Running P2PMessaging loop
```

Figure 3.35 : Factory and Farm node CLI window show silos.

Also, we can see information about any type of transactions by entering the following command followed by a type of state

```
run vaultQuery contractStateType: state-class
```



```
Thu Oct 15 02:09:26 WAT 2020>>> run vaultQuery contractStateType: supply.chain.states.Shipping
states:
- state:
    data: !<supply.chain.states.Shipping>
      source: "O=Farm, L=Bouira, C=DZ"
      destination: "O=Factory, L=Bouira, C=DZ"
      departure: 1602723476071
      arrival: 1602724100639
      capacity: 300.0
      quantity: 70.0
      temperature: 5.8
      waterRate: 86.0
      silos:
      - 1
      status: 1
      timeStamp: 1602724122189
    contract: "supply.chain.contracts.ShippingContract"
    notary: "O=Notary, L=London, C=GB"
    encumbrance: null
    constraint: !<net.corda.core.contracts.SignatureAttachmentConstraint>
      key: "aSq9DsNNvGhYxYyqA9wd2eduEAZ5AXWgJTbTEw3G5d2maAq8vtLE4kZHgCs5jcB1N31cx1hpsLeqG2ngSysVHqcXhbNts6SkRWDaV7xNcr6MtcbufGU
chxredBb6"
  ref:
    txhash: "548BB71E2D8549BBDB33AF3F3286D68FC28E661C8FDBE00BC4B580B177FB424D"
    index: 0
statesMetadata:
- ref:
    txhash: "548BB71E2D8549BBDB33AF3F3286D68FC28E661C8FDBE00BC4B580B177FB424D"
    index: 0
  contractStateClassName: "supply.chain.states.Shipping"
  recordedTime: "2020-10-15T01:08:42.993Z"
  consumedTime: null
  status: "UNCONSUMED"
  notary: "O=Notary, L=London, C=GB"
  lockId: null
  lockUpdateTime: null
  relevancyStatus: "RELEVANT"
  constraintInfo:
    constraint:
      key: "aSq9DsNNvGhYxYyqA9wd2eduEAZ5AXWgJTbTEw3G5d2maAq8vtLE4kZHgCs5jcB1N31cx1hpsLeqG2ngSysVHqcXhbNts6SkRWDaV7xNcr6MtcbufGU
chxredBb6"
totalStatesAvailable: -1
stateTypes: "UNCONSUMED"
otherResults: []
```

Figure 3.36 : Factory node CLI window show shipment transaction.

6. Performance comparison and analysis

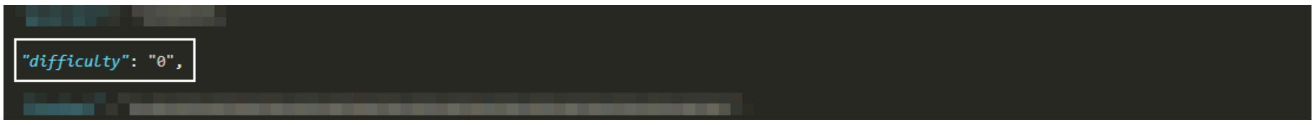
This section describes the methodology of evaluation both Ethereum and Corda and their respective settings.

6.1. Methodology and configuration

The tests was performed on two blockchain platforms, public and private, namely Ethereum and Corda, which were explained in detail in section **4** (Realization). The infrastructure where we tested is a desktop with a Ryzen 5 1600 six-core CPU, 16 GB RAM, 256 GB SSD, Windows 10 running.

For each platform, the blockchain nodes are deployed by downloading and installing software's, as Ethereum uses ***Geth*** version 1.9.19 and ***Oracle JDK*** for the Corda.

The difficulty in Ethereum Genesis Block is set to 0 (***Figure 3.37***) due to the difficulty of mining on the current machine, and the difficulty is used to determine the time it will take to mine new blocks.



```
"difficulty": "0",
```

Figure 3.37 : Configuration of difficulty in genesis block for ethereum.

Figure 3.38 represents the test architecture as there are four main modules, ***Target blockchain***, ***Pre-configuration unit***, ***Workload assessment unit***, ***Performance info scraping unit***. The first two modules are prepared before the test, and the last two are run during the test, where the evaluation workload is sent to the blockchain platform and the performance data collection module collects the transaction cases

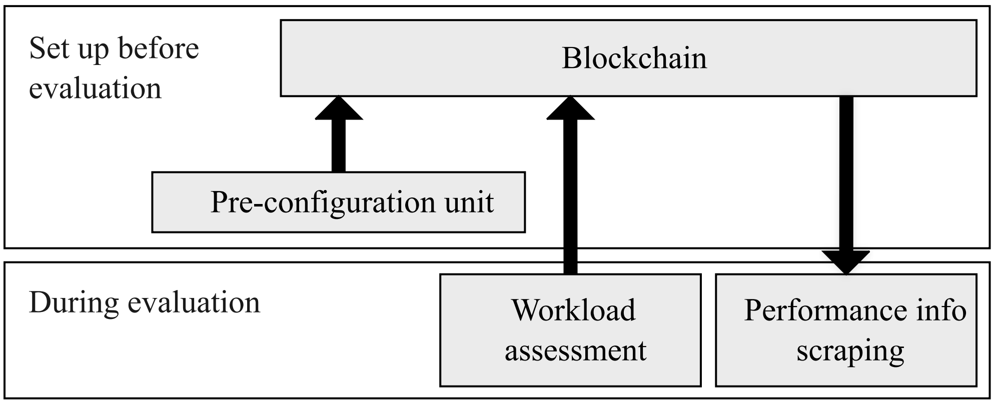


Figure 3.38 : Test architecture and units.

6.1.1. Simulation scenario

In order to evaluate the platform during the pre-formation, all smart contracts are written and published for each call-out platform during the evaluation time. We took the part of the milk production (***Milking***) mentioned in paragraph **4.6.1**, where the milk is created after verifying the health of the milked cows.

The process is performed a little different for each platform. Where the node verify its role and only let farmer to add the milked milk to the network, for Ethereum we have added a new contract that defines the role of each node in the network by the account address that is controlled by the creator of the node only and the roles are determined in advance, as for Corda, each node is given An identifier for the selected role before creating a node. The milked cows are then validated by their ***ID***, and the milk is added to the network after going through the compatibility protocol.

6.1.2. Simulation configuration

We perform several tests where many transactions are sent to the target platform with the number of ***N nodes*** and in a simultaneous way, the number of nodes (**N**) are set from **2** to **25**, and the data are simulated (such as quantity of milk, milked cows, temperature, Water rate) in a random way. In addition, the results of each test are averaged over **1000** to **2000** independent runs in each test.

Interactions between the client and the blockchain platform are accomplished by ***Jmeter***.

6.1.3. Test data collection

In order to determine the performance of blockchain platforms, the data in each transaction is calculated as follows:

Transaction deployment time (T1): is the time when transaction was deployed.

Transaction completion time (T2): is the time when transaction was confirmed by the blockchain

For Ethereum, this data was gathered directly from ***Geth CLI***, which returns the information of the transaction.

To Corda, it is done by Jmeter, where the results are extracted in a **csv** file.

6.1.4. Parameters of evaluation

The metrics picked for this evaluation are latency and throughput.

- **Latency**

  As shown in the following formula, the latency of the distributed system typically consists of three components: the time needed for the transaction request data to be transferred over the network (***TRequest***), the time required for the witness nodes to establish a consensus (***TConsensus***) and the time needed of the return of the processing result (***TResponse***) [85]

  $$DelayTransaction = TRequest + TConsensus + TResponse$$

- **Throughput**

  Transaction throughput is an important output metric for the evaluation of distributed systems. The system's throughput is typically calculated based on the number of simultaneous transactions and the amount of transactions per second (***TPS***). The time spent on a transaction is the time spent on a request from the client to the result returned by the server. The total efficiency of the system relies on the module with the lowest transaction processing capacity in the system. In a distributed system, TPS refers to the amount of transactions that the system can perform over a period of time. [85]

  $$Tps(Dt) = SumDt\ (Transactions)/Dt$$

  Where ***Dt*** would be the time was spent on processing transactions, ***SumDt(Transactions)*** means the number of transactions performed during Dt period, and ***Tps(Dt)*** refers to the amount of TPS for the Dt period.

6.2. Results and discussion

In this section, we will evaluate the performance of each blockchain platform in terms of average latency, and average throughput.

6.2.1. Latency times analysis

To test the average latency, ***1000 transactions*** were sent in each test from one node and synchronized manner, which means a transaction is sent and processed and a response is awaited and then moving to the next transaction. ***Figures 3.39, 3.40*** shows a graph of average latency which Milking transactions facing in 6 groups of tests with the error rate for both Corda and Ethereum according to network size.

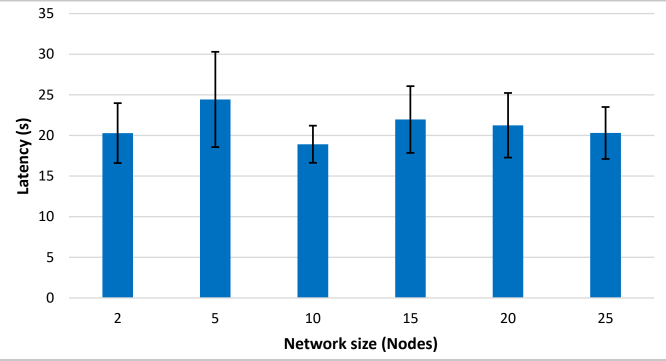
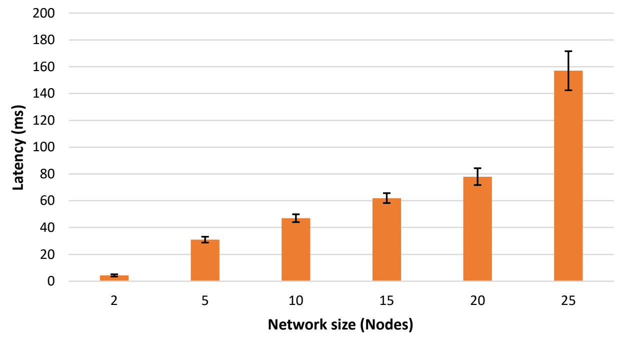
	

Figure 3.39 : Ethereum Latency with changing network size	**Figure 3.40 :** Corda Latency with changing network size

It should be mentioned that the **Y axis** represents time in a different form, as in Figure 3.40 it represents time in ***milliseconds***, while in Figure 3.39 it represents time in ***seconds***. To show the difference clearly, we add ***Figure 3.41*** bellow.

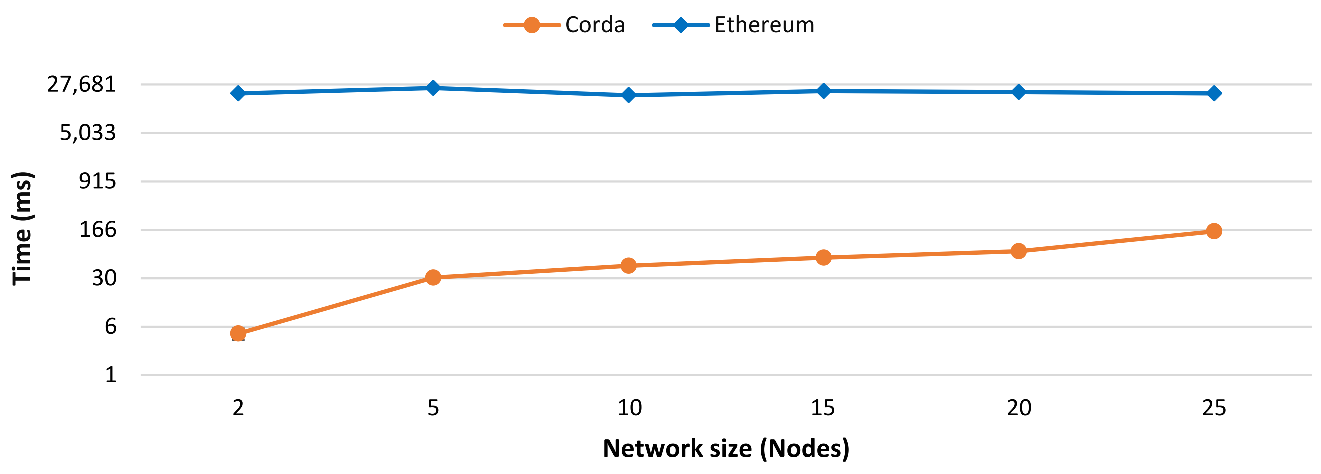


Figure 3.41 : Comparison between Ethereum and Corda latency by network size.

Discussion

As shown in **Figure 3.41**, average latency which Milking transactions facing, shows a big difference in each platform.

For a network contains two nodes, the results was very impressive in Corda as the average latency was very low, in Ethereum it was about ***20.28 sec***. Note that Ethereum latency is ***471k times*** than Corda's result in the low nodes number.

This can be related to many factors, such as **PoW** protocol which is used in Ethereum who relies on miners to add the new transaction (block) to the blockchain with a time difference which the platform sets it by adjusting the **difficulty** to maintain the time difference between blocks within ***12-30 seconds***. The difficulty in the main network at the moment of writing these lines is estimated at ***3.4P***, which is a huge number for the device on which the test was performed. The following figure shows the changes in difficulty during the month in the main network.

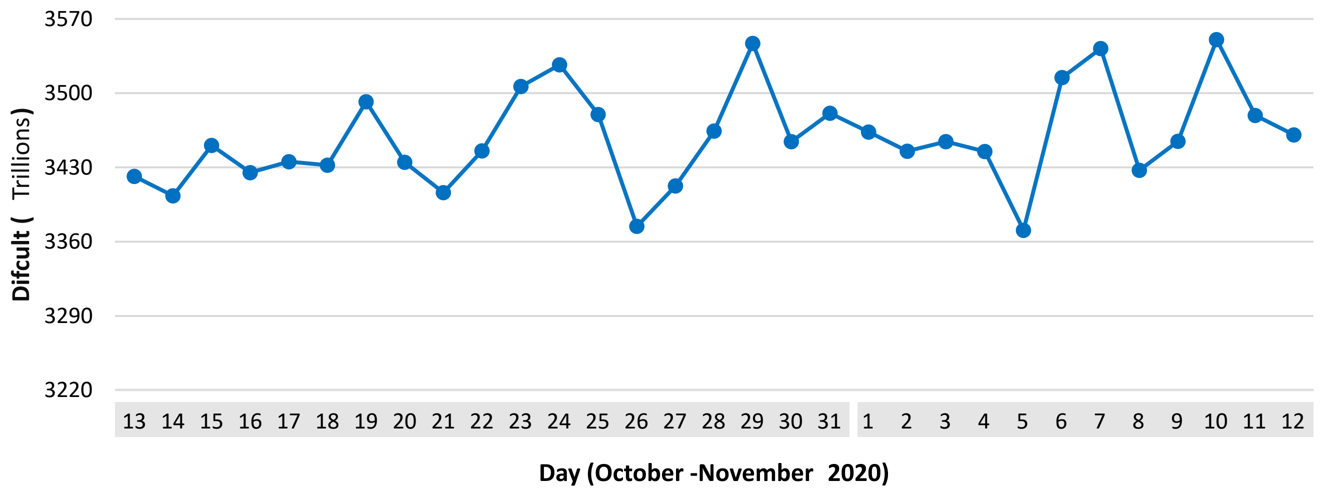


Figure 3.42 : Difficulty curve in Ethereum last month. [86]

With the increase of the number of nodes in the network (the number of nodes participating in the transaction in Corda), it can be noticed that arrive time graph of Corda rises, however it still much better than the response time of Ethereum, which continues to rise and fall unevenly in each test.

We can see the effect of network size on the average latency in Corda, unlike Ethereum which doesn't show any significant effect on the average latency and stable between ***15 and 30 seconds*** but still a high number due to relying on PoW algorithm which has long and uncertain transmission periods.

The average latency of the Ethereum isn't affected by the size of the network, which gives it great scalability, unlike the Corda, which is designed to transactions with limited parties to maintain the privacy of the network.

6.2.2. Throughput efficiency

On the other side, to test the throughput, more than ***1000 transactions*** were sent in each test in a simultaneous way, which means a request is sent and a response is awaited and then moving to the next transaction. ***Figures 3.43, 3.44*** shows a graph of average throughputs which transactions facing for both Corda and Ethereum according to network size.

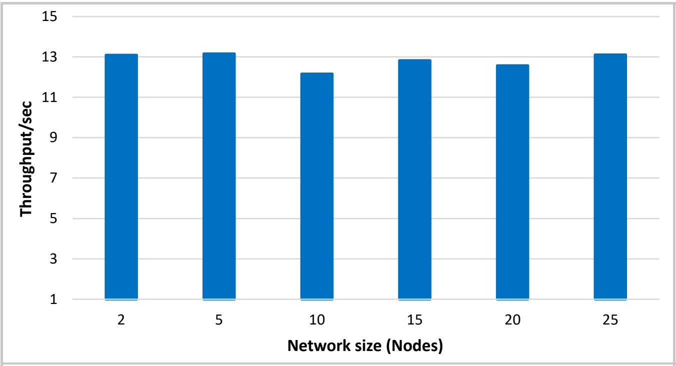
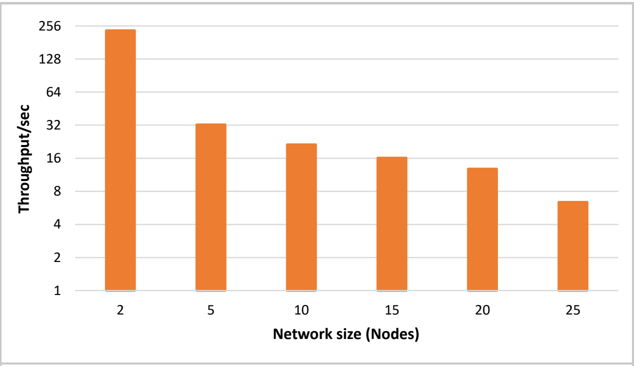
	

Figure 3.43 : Ethereum throughput with changing network size	**Figure 3.44 :**Corda throughput with changing network size

To make it easy to notice the differences in throughput for both platforms, we add ***Figure 3.45*** bellow.

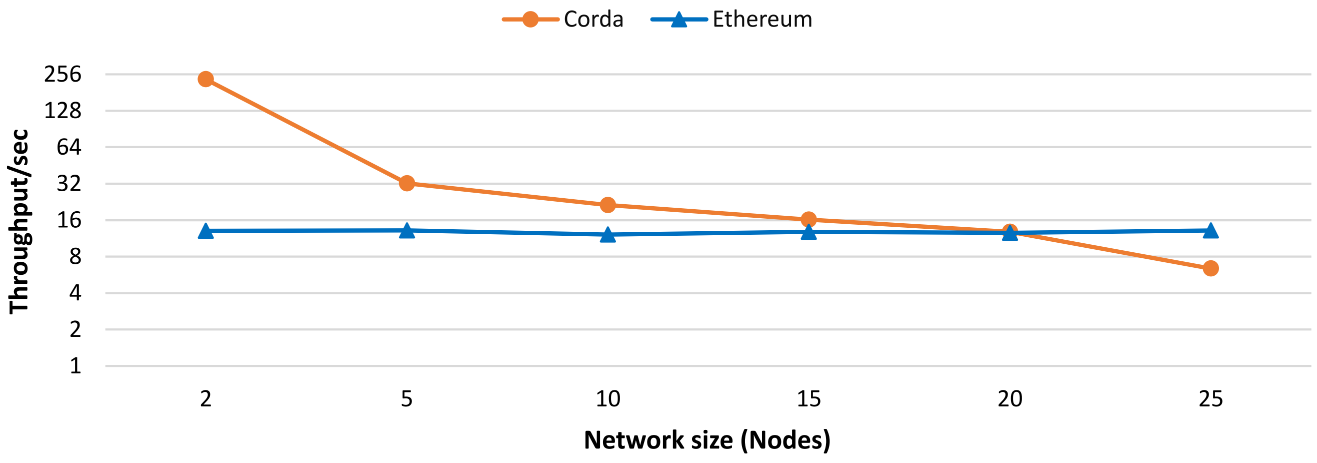


Figure 3.45 : Comparison between Ethereum and Corda throughput by network size.

Discussion

As shown in ***Figure 3.45***, for a network of ***2 nodes***, we can notice the big difference between the two platforms, as Corda gets a high average throughput that passes ***232 TPS***, in other side Ethereum platform, whose average throughput did not pass ***13 TPS***, also we noticed in low node number, throughput in Corda is about ***18 times*** of Ethereum throughput.

This can be related to many factors, such as corda's network which is similar to fully connected graph, meaning that any node in the network has the ability to communicate with any other node in the network directly (point-to-point), without the need to broadcast and wait for a response, unlike Ethereum, which relies on global broadcast and gossip networks to propagate data.

Such as the ability of Corda to implement the flows (referred to in paragraph **4.3**) in parallel to obtain significantly less response time, this lowers latency means the ability of the nodes to complete more flows at the same time and thus to reach greater throughput. And one of the positive things that made corda get this results is compressed p2p messages between nodes, which can lead to more efficient usage of network bandwidth.

We also noticed that the difference between the 2 platforms in the transfer rate decreases as we add more peers (nodes), as well we noticed that Ethereum is stabilizing in ***13 TPS***, while Corda decreases very much comparing to the first test, and it reaches about ***7 TPS*** in the last tests.

The reason that Ethereum maintain the average throughput because it relies on transaction pool, where all transactions sent it is not instantly executed, but instead becomes a **pending transaction** and is added to the **transaction pool**, the group of all transactions that have been submitted, but not yet added to a block.

The reason for the significant decrease in corda's throughput can be related to the mechanism of maintaining privacy and the mechanism of communication. Transactions in Corda affected by participants in it and the notary node which requests transactions over the network. However, the transaction recall, makes the process time consuming due to contact with the notary group and the recipient of the transaction. As indicated in the paragraph **4.3.6**

This problem is bypassed in Corda by adding many notary nodes, where the pressure is distributed on them to speed up arrival time, and from it to speed up productivity.

It can't be denied that the test itself is the main cause of the decrease, as starting 25 nodes with the limited ability of the device prevented the presentation of the actual numbers provided by the platform, which were accessed in other studies [87].

Conclusion

In this chapter we presented a analysis of how to build the dairy supply chain based on the Internet of things and blockchain technology, as we mentioned the role and importance of each of them in this supply chain, and after choosing the platform for both the public and private blockchains and mentioning the architecture of each one of them, we moved on to the construction and implementation, then we compared the two platforms after talking about the methodology used in the comparison, as this study focused on latency, throughput and the effect of network size on them by trying many tests on a local machine, and reaching an acceptable result.

General Conclusion

With the creation and continuous development of blockchain technology and the diversity that this technology is witnessing, it was necessary to make an actual comparison of a use case that defined a large activity and requires high transparency, which is the dairy supply chain in two blockchain platforms public and private, Ethereum was chosen to represent the public blockchain platform, and in return, Corda was chosen to represent the private blockchain platform, with the addition of the automation factor of Internet of Things. Several of the points mentioned above have been discussed with a detailed explanation of either two platforms or the technologies used in the development process, with an explanation of the entire phase of this study.

The results of this performance analysis study of Ethereum and Corda with a different number of transactions showed a huge gap between the two platforms, so that when networks of small sizes Corda achieves greater throughput and lower latency compared to Ethereum, but as the network expands, the gap starts to decrease as the Ethereum platform maintains the same results regardless the number of nodes. On the other hand, Corda is impacted by the increase in the number of nodes, as shows an increase in latency and a decrease in throughput, but still better in latency and worst in throughput then Ethereum results.

This decrease in the performance of Corda cannot be simply ignored as it may be gets worse in networks of large sizes, although the nodes involved in the Corda transaction do not represent the total size of the network but only the parties included in the transaction, the scenario of sharing a large number of nodes or even the entire network is possible. It should be mentioned that operating 25 nodes on a modest device prevented the actual numbers provided by the platform, meaning that it is able to give greater results in the real world. As for the Ethereum platform, which provided a steady performance throughout the tests, which gives it greater scalability for long-term expansion, with the expectation of Ethereum 2.0, which relies on the proof of stake algorithm, which will make Ethereum less dependent on computing power (PoW) and will use different algorithms to add transactions (blocks) to the blockchain, which will greatly improve the platform.

In this study, the analysis focuses on a different number of nodes, and the contribution of this study can be summarized as follows:

First, a repeatable approach is proposed for evaluating a blockchain platform. This method of performance measurement is used to test the private and public blockchain, where both of the blockchain platforms are tested with up to 1000 transactions and 25 nodes regard to throughput and latency.

Second, the results of the evaluation of these blockchain platforms and their implications are addressed, which can be taken in consideration by professionals when implementing for their own applications.

For future work, we are planning to do more tests for evaluation using different consensus protocols with the latest release of Ethereum 2.0 (PoS) and for Corda (Raft, BFT-SMaRT), with a greater number of nodes and transactions, and do the experiments on separate powerful devices to give the actual numbers. For both platforms. With using full scenario starting from the farm by adding agriculture supply chain to get more details on the cows feeding and be more automating.

Bibliography

[1]. A. Narayanan, J. Bonneau, E. Felten, A. Miller, S. Goldfeder, *"Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction, Princeton University Press",* Book, Jul 2016, Princeton University Press.

[2]. Nakamoto, *"Bitcoin: A Peer-to-Peer Electronic Cash System"*, Mar 2009, bitcoin.org/bitcoin.pdf

[3]. Block , *"Bitcoin Wiki"*, 2008. en.bitcoin.it/wiki/Block

[4]. J. Frankenfield , *"Block Header (Cryptocurrency)"*, Nov 2019, Investopedia, LLC.

[5]. J. Siska , *"Application for Demonstration of Hash Algorithms SHA-1 and SHA-2"*, 2012. OAI: oai:dspace.vutbr.cz:11012/55250

[6]. M. Arcon , *"The blockchain technology"*, University of Ljubljana, 2018. OAI: oai:repozitorij.uni-lj.si:IzpisGradiva.php?id=102682

[7]. C. Natoli, J. Yu, V. Gramoli, P. Veríssimo , *"Deconstructing Blockchains: A Comprehensive Survey on Consensus, Membership and Structure"*, Aug 2019, arXiv:1908.08316v1 [cs.DC]

[8]. Q. Mechanic, *"Bitcoin Forum. Proof of Stake instead of Proof of Work"*, Jul 2011, bitcointalk.org/index.php?topic=27787.0

[9]. Ethereum Wiki, *"Problems"*, update Oct 2020. eth.wiki/en/faqs/problems

[10]. Bytemaster, *"Transactions as proof-of-stake & the end of mining"*, Dec 2013. bitsharestalk.org/index.php?topic=1138.msg13602#msg13602A

[11]. M. Castro, B. Liskov, *"Practical Byzantine Fault Tolerance and Proactive Recovery",* 2002, MIT Laboratory for Computer Science. DOI: 10.1145/571637.571640

[12]. C. Decker, J. Seidel, R. Wattenhofer, *"Bitcoin meets strong consistency"*, in Proceedings of the 17th ICDCN, Jan 2016. DOI: 10.1145/2833312.2833321

[13]. E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, B. Ford, *"Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing"*, in 25th USENIX Security Symposium, Aug 2016, ISBN 978-1-931971-32-4

[14]. Abraham, D. Malkhi, K. Nayak, L. Ren, A. Spiegelman, *"Solida: A blockchain protocol based on reconfigurable byzantine consensus"*, in 21st International Conference on Principles of Distributed Systems, Lisbon, Portugal, 2017, DOI: 10.4230/LIPIcs.OPODIS.2017.25

[15]. R. C. Merkle, *"A digital signature based on a conventional encryption function"*, in Conference on the Theory and Application of Cryptographic Techniques, Dec 2000. DOI: 10.1007/3-540-48184-2_32

[16]. Peer-to-peer, *"P2P network and how its work"*, 2019, academy.binance.com/en/articles/peer-to-peer-networks-explained?fbclid=IwAR27VL6Y8_RWzlBlJEJSAZmj1xbCRaiqeyvKDYPPUQsVqyhcmqVNnmLpT0U

[17]. I. Bashir , *"Mastering Blockchain . 2nd ed"*, Paperback, Mar 2018, ISBN : 9781788839044

[18]. D. Drescher, *"Blockchain Basics: A Non-Technical Introduction in 25 Steps"*, Mar 2017, ISBN-10 : 1484226038

[19]. P. Baran, *"On Distributed Communications: I. Introduction to Distributed Communications Networks"*, Jan 1964, ASIN : B000TQB7JS

[20]. N. Szabo , *"Formalizing and securing relationships on public networks"*, Sep 1997, DOI: 10.5210/fm.v2i9.548

[21]. V. Buterin, *"A next-generation smart contract and decentralized application platform"*, 2015, Corpus ID: 19568665

[22]. J. Stark, *"Making sense of blockchain smart contracts"*, Updated Jun 2016, coindesk.com/making-sense-smart-contracts

[23]. V. Morabito, *"Smart contracts and licensing"*, Jan 2017, DOI: 10.1007/978-3-319-48478-5_6

[24]. K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, *"Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab"*, Aug 2016. DOI: 10.1007/978-3-662-53357-4_6

[25]. M. Liyanage, A. Manzoor, K. Thilakarathna, G. Jourjon, A. Seneviratne, *"Blockchain-based Smart Contracts Applications and Challenges"*, Jun 2019, Cite as: arXiv:1810.04699 [cs.CY].

[26]. F. Daniel, L. Guida, *"A Service-Oriented Perspective on Blockchain Smart Contracts"*, Jan 2019, DOI: 10.1109/MIC.2018.2890624

[27]. K. Christidis, M. Devetsikiotis, *"Blockchains and smart contracts for the internet of things"*, May 2016. DOI: 10.1109/ACCESS.2016.2566339

[28]. P. Esmaeilzadeh, T. Mirzaei, *"The Potential of Blockchain Technology for Health Information Exchange: Experimental Study From Patients' Perspectives"*, Jun 2019. DOI: 10.2196/14184

[29]. B. Cant, A. Khadikar, A. Ruiter, J. Bronebakk, J. Coumaros, J. Buvat, and A. Gupta, *"Smart contracts infinancial services: Getting from hype to reality"*. Capgemini Consulting, 2016. capgemini.com/consulting-de/wp-content/uploads/sites/32/2017/08/smart_contracts_paper_long_0.pdf

[30]. FinTech Network, *"Smart contracts: From ethereum to potential banking use cases"*, April 2017. blockchainapac.fintecnet.com/uploads/2/4/3/8/24384857/smart_contracts.pdf

[31]. N. Atzei, M. Bartoletti, and T. Cimoli, *"A survey of attacks on ethereum smart contracts (sok)"*, Mar 2017, DOI: 10.1007/978-3-662-54455-6_8

[32]. newsbtc, *"Danish political party may be rst to use block chain for internal voting"*, 2013, newsbtc.com/2014/04/22/danish-political-party-may-first-use-block-chain-internal-voting

[33]. P. McCorry, S. F. Shahandashti, and F. Hao, *"A smart contract for boardroom voting with maximum voter privacy"*. IACR Cryptology ePrint Archive, Jan 2017. DOI: 10.1007/978-3-319-70972-7_20

[34]. S. Nzuva, *"Smart Contracts Implementation, Applications, Benefits, and Limitations"*, Oct 2019, DOI: 10.7176/JIEA/9-5-07

[35]. P. D. Larson, A. Halldorsson, *"Logistics versus supply chain management: An international survey"*, , DOI: 10.1080/13675560310001619240

[36]. D. Yaga, P. Mell, N. Roby, K. Scarfone, *"Blockchain Technology Overview"*, Jun 2019 , DOI: 10.6028/NIST.IR.8202

[37]. M. Petersen, N. Hackius, and B. von See, *"Mapping the sea of opportunities: Blockchain in supply chain and logistics",* Oct 2018. DOI: 10.1515/itit-2017-0031

[38]. M. E. Porter and J. E. Heppelmann, *"How smart, connected products are transforming competition",* Nov 2014. hbr.org/2014/11/how-smart-connected-products-are-transforming-competition

[39]. R. Beck, M. Avital, M. Rossi, J. B. Thatcher, *"Blockchain technology in business and information systems research"*, Nov 2017. DOI: 10.1007/s12599-017-0505-1

[40]. N. Hackius, M. Petersen, *"Blockchain in logistics and supply chain: Trick or treat?"*, Oct 2017. DOI: 10.15480/882.1444

[41]. H. Diedrich, *"Ethereum: Blockchains, Digital Assets, Smart Contracts, Decentralized Autonomous Organizations"*. Lexington, KY: Wildfire Publishing, Sep 2016. ISBN-10 : 1523930470

[42]. Y. Wang, C. Huirong Chen, A. Zghari-Sales, *"Designing a blockchain enabled supply chain"*, Feb 2020, DOI: 10.1080/00207543.2020.1824086

[43]. P. M. Lourenço Costa, *"Supply Chain Management with Blockchain Technologies"*, Jul 2018, by the EUR-ACE Programme

[44]. E. Hofmann, H. Kotzab, *"A supply chain-oriented approach of working capital management"*, Sep 2010. DOI: 10.1002/j.2158-1592.2010.tb00154.x

[44]. R.R Lummus, R.J Vokurka, *"Defining Supply Chain Management: a Historical Perspective and Practical Guidelines"*, Feb 1999. DOI: 10.1108/02635579910243851

[45]. R. Ganeshan, T.P Harrison, *"Introduction to Supply Chain Management"*, 1995. DOI: 10.1007/978-1-4899-7578-2_1

[46]. M. Habib, *"Supply chain management (scm): Theory and evolution"*, Sep 2011, DOI: 10.5772/24573

[47]. M. Prokle, *"Theory and Practice of Supply Chain Synchronization"*, Sep 2017, scholarworks.umass.edu/cgi/viewcontent.cgi?article=2182&context=dissertations_2&httpsredir=1&referer=

[48]. K. Korpela, J. Hallikas, T. Dahlberg, *"Digital Supply Chain Transformation toward Blockchain Integration"*, Jan 2017, DOI: 10.24251/HICSS.2017.506

[49]. F. Isik, *"Complexity in Supply Chains : A New Approach to Quantitative Measurement of the Supply-Chain-Complexity"*, Apr 2011, DOI: 10.5772/15005

[50]. R. Wilding, *"The Supply Chain Complexity Triangle: Uncertainty Generation in the Supply Chain"*, Nov 1998. DOI: 10.1108/09600039810247524

[51]. P. M. Panayides, Y. H. Venus Lun, *"The Impact of Trust on Innovativeness and Supply Chain Performance"*, Nov 2009, DOI: 10.1016/j.ijpe.2008.12.025

[52]. J. Hoi Yan Yeung, W. Selen, M. Zhang, B. Huo, *"The Effects of Trust and Coercive Power on Supplier Integration"*, Jul 2009, DOI: 10.1016/j.ijpe.2008.07.014

[53]. CargoX, *"Reshaping the Future of Global Trade with World's First Blockchain Bill of Lading"*, 2017. cargox.io/

[54]. J. Huertas, H. Liu, S. Robinson, *"Eximchain: Supply Chain Finance Solutions on a Secured Public, Permissioned Blockchain Hybrid"*, Mar 2018. eximchain.com/Whitepaper-Eximchain.pdf

[55]. Gartner reports, *"Gartner Says 8.4 Billion Connected 'Things' Will Be in Use in 2017 Up 31 Percent From 2016, 2017"*, Feb 2017, .gartner.com/newsroom/id/3598917

[56]. B. Djellabi, M. Younis and M. Amad, *"Effective peer-to-peer design for supporting range query in Internet of Things applications"*, Jan 2020, DOI: 10.1016/j.comcom.2019.12.017

[57]. E. Okai, X. Feng, P. Sant, *"Smart Cities Survey"*, Jan 2019, DOI: 10.1109/HPCC/SmartCity/DSS.2018.00282

[58]. L. Da Xu, W. He, S. Li, *"Internet of Things in Industries: A Survey"*, Jan 2014, DOI: 10.1109/TII.2014.2300753

[59]. M. Rizqi Nur, L. Hakim, Y. Amrozi, *"Challenges In Using Blockchain For Supply Chain Management Information Systems"*, May 2020, DOI: 10.14710/jati.15.2.82-92

[60]. M. Santos, E. Moura, *"Hands-On IoT Solutions with Blockchain: Discover how converging IoT and blockchain can help you build effective solutions"*, Book, Jan 2019, ASIN : B07DTKX2YX

[62]. SigFox, *"SigFox Technology"*, sigfox.com/en/what-sigfox/technology.

[63]. Shi, Gaotao, Li , Keqiu, *"Signal Interference in WiFi and ZigBee Networks"*, Oct 2016, ASIN : B01MQ0ZWYV

[64]. D. K. Bandler, R. P. Singh, *"Dairy product"*, Update May 2018, britannica.com/topic/dairy-product

[65]. E. The Chain, *"An Investor Brief on Impacts that Drive Business Risks: DAIRY"*, Nov 2018, engagethechain.org/sites/default/files/commodity/Ceres_EngageTheChain_Dairy_0.pdf

[66]. R. Barrett, *"Wisconsin farmers forced to dump milk"*, Apr 2020, jsonline.com/story/money/2020/04/01/coronavirus-forces-dairy-farmers-dump-milk-wisconsin-covid-19/5108609002/)

[67]. Dairy, *"Dairy Management Solutions Designed for the Future"*, dairy.com

[68]. IBM, *"Walmart IBM supply chain"*, ibm.com

[69]. Kuehne + Nagel, *"K&G Logistics advancement"*, home.kuehne-nagel.com

[70]. OriginTrail, *"Utilizing Smart Sensors to Prevent Wine Fraud"*, Feb 2018, medium.com/origintrail/utilizing-smart-sensors-to-prevent-wine-fraud-origintrails-pilot-with-tagitsmart-1949dc62113f

[71]. State of the ĐApps, *"A List of 1,687 Projects Built on Ethereum"*, Jul 2018, stateofthedapps.com

[72]. J. Kelly, *"Three banks join R3 blockchain consortium taking total to 25"*, Oct 2015, uk.reuters.com/article/uk-global-banks-blockchain/three-banks-join-r3-blockchain-consortium-taking-total-to-25-idUKKCN0SM1UH20151028

[73]. *V. Buterin*, "*Ethereum: Now Going Public*", Jan 2014, blog.ethereum.org/2014/01/23/ethereum-now-going-public/

[74]. Ethereum Wiki, *"Ethereum Wiki"*, Update Jun 2020, eth.wiki

[75]. My Ether Wallet, *"Transactions. What is a Nonce?"*, kb.myetherwallet.com/en/transactions/what-is-nonce/

[76]. K. Smith, *"Ethereum's move to PoS. First version of Casper released"*, May 2018, bravenewcoin.com/insights/ethereums-move-to-pos-first-version-of-casper-released

[77]. P. Viennas, "*Ethereum Consensus and Scalability (Blockchain series. Part III)*", Oct 2018, medium.com/bethereum/ethereum-consensus-and-scalability-blockchain-series-part-iii-4acd78d0eb41

[78]. J. Kelly. *"Nine of world's biggest banks join to form blockchain partnership"*, Sep 2015, reuters.com/article/us-banks-blockchain/nine-of-worlds-biggest-banks-join-to-formblockchain-partnership-idUSKCN0RF24M20150915

[79]. R. G. Brown. *"R3 Corda: What Makes It Different"*, Oct 2016, corda.net/blog/r3-corda-what-makes-it-different/

[80]. M. Hearn, R. G. Brown, *"Corda: A distributed ledger"*. Aug 2019, r3.com/reports/corda-technical-whitepaper

[81]. D. Mohanty, *"R3 Corda for Architects and Developers"*, Book, Sep 2019, ISBN-10 : 1484245288

[82]. GEA CowScout, *"CowScout Heat Detection & Health Management"*. gea.com/en/products/milking-farming-barn/activity-detection-cowscout.jsp

[83]. Oakmedical, "*Your Worldwide Partner In The Laboratory*", oakmedical.co.za/wp-content/uploads/2018/10/Boeco_Catalog_2018.pdf

[84]. ESP8266 learning, "*SHT20 temperature and humidity sensor and ESP8266*". esp8266learning.com/sht20-temperature-and-humidity-sensor-and-esp8266.php

[85]. Y. Wu, P. Song, F. Wang, *"Hybrid Consensus Algorithm Optimization: A Mathematical Method Based on POS and PBFT and Its Application in Blockchain"*, Apr 2020, DOI: 10.1155/2020/7270624

[86]. Blockchair, "*Ethereum difficulty charts*", Nov 2020, blockchair.com/ethereum/charts/difficulty

[87]. J. Carlyle, *"Corda Performance: To infinity and beyond!"*, Apr 2018, r3.com/wp-content/uploads/2018/04/Corda-Performance-ENG.pdf

Abstract

The emergence of blockchain technology has come to enhance the independence of the Internet as a decentralized technology for managing transactions and data, which enables individuals and companies to cooperate with confidence and near-absolute transparency in the network, as an immutable ledger, the interest in this technology has increased and become covering many services, including finance, industrial fields, Internet of things, etc. many decentralized applications take advantage of this technology.

The goal of our study is to make this technology more practical, by designing and developing a blockchain-based application. The application we propose provides an efficient solution for the ordinary milk supply chain by integrating IoT solutions, However, different types of blockchain can handle supply chain issues using whether public or private blockchains. Therefore, selecting a suitable solution among these different types is deemed as the second scope of our work. We have led a performance analyses of both types of blockchain in terms of throughput and latency. Our simulations and results validation shows a clear tradeoff between both technologies, so the discussion of the results could be efficient and useful for any further relevant studies.

Keyword

Decentralization, P2P, Private blockchain, Public blockchain, Smart contracts, Supply chain, Internet of things, Corda, Ethereum.

Résumé

L'émergence de la technologie blockchain est venue renforcer l'indépendance d'Internet en tant que technologie décentralisée de gestion des transactions et des données, qui permet aux particuliers et aux entreprises de coopérer en toute confiance et avec une transparence quasi absolue dans le réseau, le considérant comme un registre immuable.L'intérêt pour cette technologie a augmenté et couvre désormais de nombreux domaines, y compris les services et domaines financiers. Industriel, Internet des Objets, etc., car de nombreuses applications bénéficient de cette technologie.

L'objectif de notre étude est de rendre cette technologie plus pratique, en concevant et développant une application basée sur la blockchain. L'application que nous proposons apporte une solution efficace à la chaîne d'approvisionnement du lait en intégrant des solutions IoT. Cependant, les différents types de blockchain (publique, privée) traitent le problème de la chaîne d'approvisionnement d'une manière différente, donc choisir le bon type parmi ces types est la deuxième partie de cette étude. Nous avons analysé les performances des deux types de blockchain en termes de débit et de latence. Comme nos simulations et la validation des résultats montrent un écart clair entre les deux techniques, la discussion des résultats peut être efficace et utile pour toute autre étude pertinente.

Mot clé

Décentralisation, P2P, Blockchain privée, Blockchain publique, Contrats intelligents, Chaîne d’approvisionnement, Internet des objets, Corda, Ethereum.

ملخص

إن ظهور تقنية البلوكشين أتى ليعزز من استقلالية الانترنت بصفتها تقنية لامركزية لإدارة المعاملات والبيانات والتي تمكن الافراد والشركات من التعاون بثقة وشفافية شبه مطلقة في الشبكة, بعتبارها كدفتر استاذ غير قابل للتغير, تزايد الاهتمام بهذه التقنية واصبحت تغطي الكثير من المجالات بما في ذالك الخدمات المالية والمجالات الصناعية و إنترنت الاشياء وما الى ذلك,حيث تستفيد العديد من التطبيقات من هذه التكنولوجيا.

الهدف من دراستنا هو جعل هذه التكنولوجيا أكثر عملية، من خلال تصميم وتطوير تطبيق قائم على البلوكشين. يوفر التطبيق الذي نقترحه حلاً فعالاً لسلسلة توريد الحليب من خلال دمج حلول إنترنت الأشياء. ومع ذلك ، تتعامل ألانواع المختلفة من البلوكشين (العامة ،الخاصة) مع مشكلة سلسلة التوريد بطريقة مختلفة، لذلك ، يعتبر اختيار النوع المناسب من بين هذه الأنواع هو الجزء الثاني لهذه الدراسة.
لقد أجرينا تحليلات لأداء كلا نوعي البلوكشين من حيث الإنتاجية ووقت الاستجابة. ،تظهر عمليات المحاكاة والتحقق من صحة النتائج لدينا تفاوت واضحة بين كلتا التقنيتين, لذلك يمكن أن تكون مناقشة النتائج فعالة ومفيدة لأية دراسات أخرى ذات صلة.

الكلمات المفتاحية

اللامركزية, الند للند, بلوكشين خاصة, بلوكشين عامة, عقود ذكية, سلسلة التوريد, انترنت الأشياء, كوردا, ايثريوم.